

PROCEEDINGS OF THE

14th INTERNATIONAL

CONFERENCE ON

DIGITAL AUDIO EFFECTS

SEPTEMBER 19-23, 2011

IRCAM, PARIS, FRANCE

<http://dafx11.ircam.fr>

Published by:

IRCAM - Centre Pompidou

<http://dafx11.ircam.fr>

<http://www.ircam.fr>

ISBN: 978-2-9540351-0-9

Credits:

Proceedings edited by Geoffroy Peeters

using L^AT_EX's 'confproc' package, version 0.7 by V. Verfaillie

Cover design: BelleVille 2011

Logo photo: Geoffroy Peeters

Printed in Paris City by Présence Graphique — September 2011

All copyrights remain with the authors

DAFx-11 is supported by:

IRCAM - Centre Pompidou

<http://www.ircam.fr>



Ministère de la Culture et Communication

<http://www.culture.gouv.fr/>



Centre National de la Recherche Scientifique

<http://www.cnrs.fr/>



Université Pierre et Marie Curie

<http://www.upmc.fr/>

Preface

Welcome to DAFx-11

On behalf of the Local Organizing Committee, we would like to welcome you to the 14th conference on Digital Audio Effects, DAFx-11, which is organized and hosted by the Institut de Recherche et Coordination Acoustique/ Musique (IRCAM), Centre Pompidou in Paris, France. The DAFx conference is an international forum for the presentation of technological advances and research results in the field of digital audio effects and related disciplines. It brings together leading researchers, engineers, and developers from around the world.

DAFx-11 is structured around three full days of conferences preceded by an afternoon of 3 tutorials and followed by a full day of 3 satellite workshops providing in-depth treatment of specific topics. The three-day conference consists of technical paper and invited keynote talks. 69 papers have been selected organized in 33 oral presentations (12 sessions), 33 poster presentations (9 sessions), 3 State-of-the-Art (STAR) papers and 3 keynote talks. The STAR paper is a new category of submission that aims at providing profound review of recent research in particular research fields. Due to the many submissions related to audio signal processing (audio representation, modeling, coding, synthesis, and transformation), 5 oral sessions are dedicated to these topics. Traditional key issues of DAFx, such as the capture, analysis of audio signals, the synthesis of acoustic fields/ spatial sounds, virtual analog models, physical models and virtual musical instruments are represented in 7 oral sessions. Among these topics, audio indexing and sonic interaction are the focus of many contributions. In addition to the 12 oral sessions, 9 poster sessions will be held. In DAFx-11, poster sessions do not have a specific topic. We intentionally mixed the contributions inside each session in order to favour exchanges among researchers of a given field and continuous interest of the audience. In order to better highlight posters, each poster author is welcome to provide a short oral presentation at the end of the preceding oral session.

The program committee invited three keynote speakers, which will open each conference day in the morning. Udo Zölzer, one of the founder of DAFx, will discuss Pitch-based Digital Audio Effects. David Zicarelli, founder of Cycling 74 (Max/MSP), will discuss the advantages and disadvantages of working on algorithms for digital signal processing in a visual way and he will present recent works in visual editing of generalized synchronous DSP graphs, domain-specific graphs, visual performance monitoring, and filter design. Patrick Flandrin will present and illustrate some recent methodological advances in spectral processing, from wavelet-like transforms to sparse time-frequency distributions and oscillations-based empirical mode decompositions 200 years after Fourier published his fundamental essay on heat diffusion.

Complementing the scientific program, DAFx-11 also offers a full program of social events, providing a brief overview of the numerous IRCAM activities in audio research and music production. It includes the world premiere of composer Andrea Agostini "Electrons libres", an IRCAM production commissioned by the Musée du Louvre, presentations by some of IRCAM research groups and a demonstration of the wave field synthesis system at IRCAM. The social events also include a visit of the National Museum of Modern Art (Centre Pompidou) and a charming gala dinner on a riverboat that will sail up and down the Seine.

We thank all the authors for submitting their works, all the local scientific committee members and reviewers ensuring a high quality of the scientific program of DAFx-11. Thanks to our sponsors: the Centre National de la Recherche Scientifique (CNRS), the French Ministry of Culture and Communication and the Centre Pompidou. We would also like to thank the organizers of the DAFx-10 conference, Alois Sontacchi, Franz Zotter and Hannes Pomperger as well as Udo Zölzer kindly sharing their practical experience in organizing a DAFx-conference with us. We also want to thank all other helping hands that made the organization of DAFx-11 possible: Thank you!

The members of the DAFx-11 organizing committee wish you a successful meeting, accompanied by many fruitful discussions, experiences, exchanges, and new contacts. We hope that this year's DAFx conference raises more challenging questions than it answers. Enjoy Paris!

Paris, September 7, 2011 - **Geoffroy Peeters, Markus Noisternig, Olivier Warusfel** - DAFx-11 Chairs

DAFX-11 Local Organizing Committee

Geoffroy	Peeters	Conference and paper chairs
Markus	Noisternig	
Olivier	Warusfel	

Sylvie	Benoit	Coordination
Samuel	Goldszmidt	Web
Stéphanie	Leroy	Registration
Claire	Marquet	Communication
Murielle	Ducas	

DAFx-2011 Local Scientific Committee

Christophe	d'Alessandro	LIMSI/ CNRS
Laurent	Daudet	ESPCI
Emmanuel	Favreau	INA/ GRM
Thomas	Hélie	CNRS
Markus	Noisternig	IRCAM/ CNRS
Geoffroy	Peeters	IRCAM/ CNRS
Clara	Suied	ENS/ FPGG
Olivier	Warusfel	IRCAM/ CNRS

DAFx Conferences

DAFx			Main Web Page	http://www.dafx.de/
DAFx 1998	Barcelona	Spain	November 19-21, 1998	http://iua.upf.edu/dafx98/
DAFx 1999	Trondheim	Norway	December 9-11, 1999	
DAFx 2000	Verona	Italy	December 7-9, 2000	http://profs.sci.univr.it/~dafx/
DAFx 2001	Limerick	Ireland	December 6-8, 2001	http://www.csis.ul.ie/dafx01/
DAFx 2002	Hamburg	Germany	September 26-28, 2002	http://www2.hsu-hh.de/ant/dafx2002/
DAFx 2003	London	United Kingdom	September 8-11, 2003	http://www.elec.qmul.ac.uk/dafx03/
DAFx 2004	Naples	Italy	October 5-8, 2004	http://dafx04.na.infn.it/
DAFx 2005	Madrid	Spain	September 20-22, 2005	
DAFx 2006	Montreal	Canada	September 18-20, 2006	http://www.dafx.ca/
DAFx 2007	Bordeaux	France	September 10-15, 2007	http://dafx.labri.fr/main/
DAFx 2008	Espoo	Finland	September 1-4, 2008	http://www.acoustics.hut.fi/dafx08/
DAFx 2009	Como	Italy	September 1-4, 2009	http://dafx09.como.polimi.it/
DAFx 2010	Graz	Austria	September 6-10, 2010	http://dafx10.iem.at/
DAFx 2011	Paris	France	September 19-23, 2011	http://dafx11.ircam.fr/
DAFx 2012	York	United Kingdom		
DAFx 2013	Maynooth	Ireland		
DAFx 2014	Erlangen	Germany		

Notes on the Review Process

The call for contributions to DAFx-11 was published in December 2010. The call concerned several categories of contributions: research papers, State-of-the-Art (STAR) papers, tutorials and satellite workshops. The newly introduced STAR paper category is a 12-page paper associated with a 30-minutes oral presentation. Its aim is to provide a comprehensive overview of the current state of research in a given DAFx topic. The STAR papers were reviewed like the other types of submissions but had to provide a deep, full and fair review of existing works. Their number was limited to three.

DAFx-11 received over 100 submissions from which 69 submissions have been selected and assigned to 12 oral and 9 poster sessions: 3 STAR papers, 33 oral presentations and 33 poster presentations. The rejection rate of the DAFx-11 conference is therefore around 30%.

DAFx-11 introduced a new reviewing process. In order to allow a better assignment of papers to reviewers and therefore better reviews, a local scientific committee (LSC) composed of nine senior researchers whose expert knowledge covers at least one of DAFx topics has been composed. A first LSC meeting was held to assign each paper to one of the LSC members according to its addressed topic(s). The PC members pre-reviewed the submissions before assigning them to three international reviewers. Consequently, not only title, keywords and topics but also the content of a submission have been taken into account for the paper assignment procedure. This procedure further helps to avoid possible conflicts of interest. Most of papers were reviewed by three independent reviewers, among which at least one DAFx Board member (when possible), one previous DAFx participant (in order to preserve continuity over years of DAFx program and spirit) and one extra specialist. Hence, more than 100 different reviewers participated to the reviewing process.

A second LSC meeting was then held to discuss all acceptance decisions in detail. Submitted papers were ranked according to their average reviewing score. Decision on their acceptance was based on this ranking and, in case of contradictory reviews, on reviewing comments. For the assignment to oral or poster sessions, the LSC took also into account specific requests of the authors, recommendations of reviewers, and the interest of the paper for DAFx general audience. It should be noted that the length of the final paper was not systematically taken as a criterion for orienting the presentation to oral or poster sessions. The DAFx co-chairs agreed on a non-restrictive policy on the lengths of the submitted papers in order to avoid large modifications of the papers between the reviewing process and the publication process. The authors have been encouraged to revise their manuscripts according to the comments of the reviewers and LCS members. The revised papers are published in the DAFx conference proceedings.

The reviewing process was done using the OpenConf (free version) system (and ad-hoc Matlab and Python scripts).

We would like to thank all the reviewers and authors for their tremendous work and for shaping this conference. We are very grateful to the Local Scientific Committee members for their dedication and efforts to improve the scientific quality and research value of DAFx-11.

Satellite workshops and tutorials were selected directly by the DAFx organizing committee. The selection of satellite workshops was based on the abstracts submitted and personal talks with the workshop organizers. They comprise physical modeling (Modalys), audio-graphic virtual environments, and Computational Auditory Scene Analysis (CASA). The main criterion for the selection of tutorials was "inter-disciplinarity". The three selected tutorials present new mathematical tools which are applicable to various research fields, such as blind source separation (BSS), Volterra series expansion, and geometric information.

DAFx Board

Daniel	Arfib	CNRS-LMA, France
Nicola	Bernardini	Cons. Cesare Pollini Padova, Italy
F. Javier	Casajús	ETSI Telecomunicacion-UPM, Spain
Laurent	Daudet	Univ. Paris Diderot, Paris 7, France
Philippe	Depalle	McGill University, Canada
Giovanni	De Poli	Dept. Information Engineering, Univ. of Padova, Italy
Myriam	Desainte-Catherine	LaBRI, Univ. Bordeaux, France
Markus	Erne	Scopein Research, Aarau, Switzerland
Gianpaolo	Evangelista	Linköping Univ., Sweden
Emmanuel	Favreau	INA-GRM, Paris, France
Simon	Godsill	Univ. of Cambridge, UK
Robert	Höldrigh	IEM, Univ. of Music and Performing Arts Graz, Austria
Pierre	Hanna	Univ. Bordeaux 1, France
Jean-Marc	Jot	DTS, CA, USA
Victor	Lazzarini	National University of Ireland, Maynooth
Sylvain	Marchand	LaBRI, Univ. Bordeaux 1, France
Damian	Murphy	Univ. of York, UK
Søren	Nielsen	SoundFocus, Aarhus, Denmark
Markus	Noisternig	IRCAM, Paris, France
Luis	Ortiz Berenguer	U.P. Madrid, Spain
Geoffroy	Peeters	IRCAM, Paris, France
Rudolf	Rabenstein	Univ. Erlangen-Nuremberg, Germany
Davide	Rocchesso	IUAV Univ. Venice, Italy
Jøran	Rudi	NoTAM, Oslo, Norway
Mark	Sandler	Centre for Digital Music, Queen Mary Univ. London, UK
Augusto	Sarti	DEI, Politecnico di Milano, Italy
Lauri	Savioja	TKK, Espoo, Finland
Xavier	Serra	UPF, Barcelona, Spain
Julius O.	Smith	CCRMA, Stanford Univ., USA
Alois	Sontacchi	IEM, Univ. of Music and Performing Arts Graz, Austria
Marco	Tgliasacchi	DEI, Politecnico di Milano, Italy
Todor	Todoroff	ARTEM, Bruxelles, Belgium
Jan	Tro	NTNU, Trondheim, Norway
Vesa	Välimäki	TKK, Espoo, Finland
Udo	Zölzer	Helmut-Schmidt Univ., Hamburg, Germany

DAFx-11 Reviewer Board

Jonathan	Abel	Jonathan	Le Roux
Kamil	Adiloglu	Heidi-Maria	Lehtonen
Federico	Avanzini	Pierre	Leveau
Roland	Badeau	Tapio	Lokki
Balázs	Bank	Piotr	Majdak
Gerald	Thomas Beauregard	Sylvain	Marchand
Juan	Bello	Rémi	Mignot
Joël	Bensoam	Nicolas	Misdariis
Nicola	Bernardini	Meinard	Mueller
Stephanie	Bertet	Damian	Murphy
Frédéric	Bimbot	Gautham	Mysore
Jean	Bresson	Thibaud	Necciari
Thibaut	Carpentier	Sören	Nielsen
René	Caussé	Markus	Noisternig
Gérard	Chollet	Nobutaka	Ono
Ivan	Cohen	Jyri	Pakarinen
Christophe	d' Alessandro	Jouni	Paulus
Nicolas	d' Alessandro	Geoffroy	Peeters
Laurent	Daudet	Henri	Penttinen
Bertrand	David	Laurent	Pottier
Giovanni	De Poli	Daniel	Pressnitzer
Philippe	Depalle	Rudolf	Rabenstein
Olivier	Derrien	Jean-Bernard	Rault
Myriam	Desainte-Catherine	Emmanuel	Ravelli
Boris	Doval	Marc	Rébillat
Angélique	Drémeau	Josh	Reiss
Thierry	Dutoit	Gaël	Richard
Gianpaolo	Evangelista	Albert	Rilliard
Benoît	Fabre	Curtis	Roads
Emmanuel	Favreau	Axel	Roebel
Cédric	Févotte	Lauri	Savioja
Afroditi	Filippas	Andrew	Schmeder
Thomas	Fillon	Norbert	Schnell
Patrick	Flandrin	Diemo	Schwarz
Federico	Fontana	Shihab	Shamma
Hugues	Genevois	Julius	Smith
Laurent	Girin	Alois	Sontacchi
Volker	Gnann	Bob	Sturm
Fabien	Gouyon	Nicolas	Sturmel
Rémi	Gribonval	Clara	Suied
Pierre	Hanna	Damien	Tardieu
Thomas	Hélie	Stéphan	Tassart
Romain	Hennequin	Jan	Tro
Perfecto	Herrera	Nicolas	Tsingos
Robert	Hoeldrich	Vesa	Valimaki
Jean-Marc	Jot	Maarten	van Walstijn
Brian	Katz	Tuomas	Virtanen
Matthieu	Kowalski	Olivier	Warusfel
Pierre	Lanchantin	Chunghsin	Yeh
Olivier	Lartillot	David	Zicarelli
Victor	Lazzarini	Udo	Zoelzer
Sylvain	Le Beux	Franz	Zotter
Jean-Loic	Le Carrou		

Conference Program

Preface	iii
Monday 2011/09/19 PM	1
Tutorial 1: "Introduction to Volterra series and applications to physical audio signal processing" - Thomas Hélie	1
Tutorial 2: "Music Source Separation" - Emmanuel Vincent	1
Tutorial 3: "Applications of Information Geometry to Audio Signal Processing" - Arshia Cont, Arnaud Dessein	1
Tuesday 2011/09/20 AM	1
Keynote 1: Udo Zölzer - "Pitch-based Digital Audio Effects"	1
Oral Session 1: Capture, and Analysis of Audio Signals	1
2 Microphone Interference Reduction in Live Sound <i>Alice Clifford, Josh Reiss</i>	
11 The Image-Source Reverberation Model in an N-Dimensional Space <i>Stephen McGovern</i>	
Poster Session 1	19
19 Computationally Efficient Hammond Organ Synthesis <i>Jussi Pekonen, Tapani Pihlajamäki, Vesa Välimäki</i>	
23 Structured sparsity for audio signals <i>Kai Siedenburg, Monika Dörfler</i>	
27 Modeling of the Carbon Microphone Nonlinearity for a Vintage Telephone Sound Effect <i>Sami Oksanen, Vesa Välimäki</i>	
Oral Session 2: Virtual Analog Models	31
31 Physical Modelling of a Wah-wah Effect Pedal as a Case Study for Application of the Nodal DK Method to Circuits with Variable Parts <i>Martin Holters, Udo Zölzer</i>	
37 Automated Calibration of a Parametric Spring Reverb Model <i>Hannes Gamper, Julian Parker, Vesa Välimäki</i>	
45 Lyapunov Stability Analysis of the Moog Ladder Filter and Dissipativity Aspects in Numerical Solutions <i>Thomas Hélie</i>	
Poster Session 2	53
53 A Preliminary Study on Sound Delivery Methods for Footstep Sounds <i>Luca Turchet, Stefania Serafin</i>	
59 Simulation of a Vacuum-Tube Push-Pull Guitar Power Amplifier <i>Jaromir Macak, Jiri Schimmel</i>	
63 Analysis and Trans-synthesis of Acoustic Bowed-String Instrument Recordings: a Case Study using Bach Cello Suites <i>Yin-Lin Chen, Tien-Ming Wang, Wei-Hsiang Liao, Alvin Wen-Yu Su</i>	
69 DHM and FDTD based Hardware Sound Field Simulation Acceleration <i>Yasushi Inoguchi, Tan Yiyu, Yukinori Sato, Makoto Otani, Yukio Iwaya</i>	

- 73 Sinusoid Extraction and Saliency Function Design for Predominant Melody Estimation
Justin Salamon, Emilia Gómez, Jordi Bonada

Tuesday 2011/09/20 PM 81

Star Paper 1 81

- 81 Sparse Atomic Modeling of Audio: a Review
Corey Kereliuk, Philippe Depalle

Oral Session 3: Audio Representation / Modeling 93

- 93 Constructing an invertible constant-Q transform with nonstationary Gabor frames
Gino Angelo Velasco, Nicki Holighaus, Monika Dörfler, Thomas Grill
- 101 Multiresolution STFT Phase Estimation with Frame-Wise Posterior Window-Length Decision
Volker Gnann, Martin Spiertz
- 107 Sound Analysis and Synthesis Adaptive in Time and Two Frequency Bands
Marco Liuni, Peter Balazs, Axel Röbel

Poster Session 3 115

- 115 Exponential Frequency Modulation Bandwidth Criterion for Virtual Analog Applications
Joseph Timoney, Victor Lazzarini
- 119 Towards Ontological Representations of Digital Audio Effects
Thomas Wilmering, György Fazekas, Mark B. Sandler
- 123 Identification of Time-frequency Maps for sounds timbre discrimination
Olivero Anaik

Oral Session 4: Audio Indexing (Audio-Based Music Information Retrieval Systems) 127

- 127 Combining classifications based on local and global features: application to singer identification
Lise Régnier, Geoffroy Peeters
- 135 Enhanced Beat Tracking with Context-Aware Neural Networks
Sebastian Böck, Markus Schedl
- 141 On the Use of Perceptual Properties for Melody Estimation
Wei-Hsiang Liao, Alvin Wen-Yu Su, Chunghsin Yeh, Axel Röbel

Wednesday 2011/09/21 AM 147

Keynote 2: David Zicarrelì - "Recent developments in signal processing editing and visualization" 147

Oral Session 5: Audio Coding / Transmission 147

- 148 Black Box methodology for the characterization of Sample Rate Conversion systems
Stéphan Tassart
- 155 Optimal Filter Partitions for Real-Time FIR Filtering using Uniformly-Partitioned FFT-based Convolution in the Frequency-Domain
Frank Wefers, Michael Vorländer

Poster Session 4 163

- 163 A Simple Digital Model of the Diode-Based Ring-Modulator
Julian Parker
- 167 Gestural Auditory and Visual Interactive Platform
Baptiste Caramiaux, Sarah Fdili Alaoui, Tifanie Bouchara, Gaetan Parseihian, Marc Rébillat

- 171 Time-Variant Delay Effects based on Recurrence Plots
Taemin Cho, Jon Forsyth, Laewoo Kang, Juan Bello

Oral Session 6: Capture, Analysis, and Synthesis of Acoustic Fields/ Spatial Sounds

177

- 177 A Sound Localization based Interface for Real-Time Control of Audio Processing
Daniele Salvati, Sergio Canazza, Antonio Rodà
- 185 Similarity-based Sound Source Localization with a Coincident Microphone Array
Karl Freiburger, Alois Sontacchi
- 191 A Comparison of Analysis and Resynthesis Methods for Directional Segmentation of Stereo Audio
Jeremy Wells

Poster Session 5

199

- 199 FAUST-STK: a set of linear and nonlinear physical models for the FAUST programming language
Romain Michon, Julius Orion III Smith
- 205 Analysis and Simulation of an Analog Guitar Compressor
Oliver Kröning, Kristjan Dempwolf, Udo Zölzer
- 209 A Single-Azimuth Pinna-Related Transfer Function Database
Simone Spagnol, Marko Hiipakka, Ville Pulkki
- 213 FAUST Architectures Design and OSC Support.
Dominique Fober, Yann Orlarey, Stéphane Letz
- 217 A Grammar for Analyzing and Optimizing Audio Graphs
Vesa Norilo

Wednesday 2011/09/21 PM

221

Star Paper 2

221

- 221 State of the Art in Sound Texture Synthesis
Diemo Schwarz

Oral Session 7: Audio Synthesis / Transformation

233

- 233 Vector Phaseshaping Synthesis
Jari Kleimola, Victor Lazzarini, Joseph Timoney, Vesa Välimäki
- 241 Non-Parallel Singing-Voice Conversion by Phoneme-based Mapping and Covariance Approximation
Fernando Villavicencio, Hideki Kenmochi
- 249 Application of non-negative matrix factorization to signal-adaptive audio effects
Ryan Sarver, Anssi Klapuri

Poster Session 6

253

- 253 Block Processing Strategies for Computationally Efficient Dynamic Range Controllers
German Ramos
- 257 A Physically-motivated Triode Model for Circuit Simulations
Kristjan Dempwolf, Udo Zölzer
- 265 A Simple and Efficient Fader Estimator for Broadcast Radio Unmixing
Mathieu Ramona, Gaël Richard

Oral Session 8: Audio Synthesis / Transformation

269

- 269 PVSOLA: A Phase Vocoder with Synchronized OverLap-Add
Alexis Moinet, Thierry Dutoit

- 277 Vivos Voco: A survey of recent research on voice transformations at IRCAM
Pierre Lanchantin, Snorre Farner, Christophe Veaux, Gilles Degottex, Nicolas Obin, Greg Beller, Fernando Villavicencio, Stephan Huber, Geoffroy Peeters, Axel Röbel, Xavier Rodet
- 287 Efficient Polynomial Implementation of the EMS VCS3 Filter Model
Stefano Zambon, Federico Fontana

Thursday 2011/09/22 AM 291

Keynote 3: Patrick Flandrin - "Fourier + 200" 291

Oral Session 9: Interaction 291

- 292 Interaction-optimized Sound Database Representation
Ianis Lallemand, Diemo Schwarz
- 301 Realtime system for backing vocal harmonization
Adrian von dem Knesebeck, Sebastian Kraft, Udo Zölzer

Poster Session 7 307

- 307 Phantom Source Widening With Deterministic Frequency Dependent Time Delays
Franz Zotter, Matthias Frank, Georgios Marentakis, Alois Sontacchi
- 313 Implementing Real-Time Partitioned Convolution Algorithms on Conventional Operating Systems
Eric Battenberg, Rimas Avizienis
- 321 Transforming Vibrato Extend in Monophonic Sounds
Axel Röbel, Simon Maller, Javier Contreras

Oral Session 10: Physical Models and Virtual Musical Instruments 329

- 329 Harpsichord Sound Synthesis using a Physical Plectrum Model Interfaced with the Digital Waveguide
Chao-Yu Jack Perng, Julius Smith, Thomas Rossing
- 337 Modelling of Brass Instrument Valves
Stefan Bilbao
- 345 Physical Model of the String-Fret Interaction
Gianpaolo Evangelista

Poster Session 8 353

- 353 A High-Rate Data Hiding Technique for Audio Signals based on INTMDCT Quantization
Jonathan Pinel, Laurent Girin
- 357 Mapping blowing pressure and sound features in recorder playing
Leny Venceslas, Francisco Garcia, Alfonso Pérez, Esteban Maestre
- 361 Nonlinear Allpass Ladder Filters in FAUST
Julius Smith, Romain Michon
- 365 A Csound Opcode for a Triode Stage of a Vacuum Tube Amplifier
Marco Fink, Rudolf Rabenstein
- 371 Generalized Reassignment with an Adaptive Polynomial-Phase Fourier Kernel for the Estimation of Non-Stationary Sinusoidal Parameters
Saso Musevic, Jordi Bonada

Thursday 2011/09/22 PM 375

Star Paper 3 375

- 375 Signal Reconstruction from STFT magnitude : a State of the Art
Nicolas Sturm, Laurent Daudet

Oral Session 11: Audio Representation / Modeling

387

- 387 Modal analysis of impact sounds with ESPRIT in Gabor transforms
Adrien Sirdey, Olivier Derrien, Richard Kronland-Martinet, Mitsuko Aramaki
- 393 A Parametric Model of Piano Tuning
François Rigaud, Bertrand David, Laurent Daudet
- 401 Audio De-Thumping using Huang's Empirical Mode Decomposition
Paulo Antonio Andrade Esquef, Guilherme Sausen Welter

Poster Session 9

409

- 409 Generation of Non-repetitive Everyday Impact Sounds for Interactive Applications
Wasim Ahmad, Ahmet Kondo
- 417 Synchronization of intonation adjustments in violin duets: towards an objective evaluation of musical interaction
Panagiotis Papiotis, Esteban Maestre, Marco Marchini, Alfonso Pérez
- 425 GMM supervector for Content Based Music Similarity
Christophe Charbuillet, Damien Tardieu, Geoffroy Peeters

Oral Session 12: Audio Indexing (Audio-Based Music Information Retrieval Systems)

429

- 429 Automatic Alignment of Audio Occurrences: Application to the Verification and Synchronization of Audio Fingerprinting Annotation
Mathieu Ramona, Geoffroy Peeters
- 437 Multi-Probe Histograms: A Mid-Level Harmonic Feature for Music Structure Segmentation
Kaiser Florian, Thomas Sikora
- 441 Production Effect: Audio Features for Recording Techniques Description and Decade Prediction
Damien Tardieu, Emmanuel Deruty, Christophe Charbuillet, Geoffroy Peeters

Friday 2011/09/23 AM

447

Satellite Workshop 1: "Versatile Sound Models for Interaction in Audio-Graphic Virtual Environments: Control of Audio-graphic Sound Synthesis" - Roland Cahen, Diemo Schwarz, Hui Ding

447

Satellite Workshop 2 : "Modalys, a physical synthesizer: more than twenty years of researches developments and musical uses" - Nicolas Ellis, Joel Bensoam, Jean Lochard, René Caussé

447

Friday 2011/09/23 PM

447

Satellite Workshop 3: "From ASA to CASA, what does the C stand for anyway?" - Mathieu Lagrange, Luis Gustavo Martins

447

Index of Authors

449

Tutorial 1 - Introduction to Volterra series and applications to physical audio signal processing - *Thomas H  lie*

A Volterra series is an input-to-output representation which is adapted to dynamical systems including some analytic nonlinearities. It extends notions of linear filtering: the "impulse response" and the "transfer function" are generalized into multi-variate "convolution kernels" and "transfer kernels", respectively. These kernels isolate and sort the linear, quadratic, cubic (etc) homogeneous contributions of the dynamics. In this tutorial, Volterra series, their basic properties and their links with standard linear tools are presented. A practical method to solve nonlinear differential problems by using Volterra series is proposed in two steps. It sequentially answers to the following questions: (1) How to derive the transfer kernels for a given problem? (2) How to build a realization and a simulation from the transfer kernels? Then, applications on audio and acoustical problems are presented. Finally, some computable results on convergence domains and guaranteed error bounds are given.

Tutorial 2 - Music Source Separation - *Emmanuel Vincent*

Source separation consists of extracting the signal produced by each sound source from a recording. It is a mainstream topic in music and audio processing, with applications ranging from speech enhancement and automatic speech recognition to 3D music upmixing and post-production. In this tutorial, I will present the sound cues which can be exploited for source separation and explain how they translate into three main paradigms: computational auditory scene analysis, probabilistic linear modeling and probabilistic variance modeling. I will give example algorithms for each paradigm, including the popular ICA and NMF algorithms, and illustrate the performance via a number of sound examples. Finally, I will show that the latter paradigm leads to a flexible audio source separation framework able to jointly exploit a wide range of prior information about the sources.

Tutorial 3 - Introduction to Volterra series and applications to physical audio signal processing - *Arshia Cont, Arnaud Dessein*

In this tutorial, we present some applications of information geometry to audio signal processing. In general terms, information geometry is a field of mathematics that studies the notions of probability and of information by the way of differential geometry. This provides a comprehensive framework that allows to quantify, process and represent the information contained in audio signals. We focus on the computational aspects of information geometry, and discuss generic tools to deal with exponential families which encompass most of the distributions commonly used in statistical learning. Moreover, exponential families possess a canonical dually flat geometry which generalizes the standard self-dual Euclidean geometry, with two dual Bregman divergences instead of the self-dual Euclidean distance, as well as dual geodesics, a generalized Pythagorean theorem and dual projections. We demonstrate a Matlab toolbox implementing several machine learning algorithms that have been recently generalized to these geometries, such as centroid computation and hard clustering (k-means), parameter estimation and soft clustering (expectation-maximization), proximity queries in ball trees (nearest-neighbors search, range search). We show some applications to audio processing, in particular to segmentation into quasi-stationary chunks that form consistent informative entities. These entities can then be treated as symbols for applications such as music similarity analysis, musical structure discovery, query by similarity, audio recombination by concatenative synthesis, and computer-assisted improvisation.

Keynote 1 - Pitch-based Digital Audio Effects - *Udo Z  lzer*

Digital audio effects are usually controlled by certain parameters of users and the incoming audio signal. The combination of user defined parameters and signal adaptive parameters leads to more exiting audio effects where the main effect parameters change according the audio input. The talk will cover several pitch-based audio effects, will discuss detection algorithms and special effect realizations.

Udo Z  lzer received the Diplom-Ingenieur degree in electrical engineering from the University of Paderborn in 1985, the Dr.-Ingenieur degree from the Technical University Hamburg-Harburg (TUHH) in 1989 and completed a Habilitation in communications engineering at the TUHH in 1997. Since 1999 he has been a professor and head of the Department of Signal Processing and Communications at the Helmut Schmidt University & University of the Federal Armed Forces in Hamburg, Germany. His research interests are audio and video signal processing and communication.

MICROPHONE INTERFERENCE REDUCTION IN LIVE SOUND

Alice Clifford, Josh Reiss*

Centre for Digital Music
Queen Mary, University of London
London, UK

alice.clifford@eecs.qmul.ac.uk

ABSTRACT

When multiple microphones are used to reproduce multiple sources microphone interference, or bleed, can occur due to each microphone picking up more than one source. This paper proposes combining the crosstalk resistant adaptive noise canceller (CTRANC) algorithm with centred adaptive filters using an estimation of delay to suppress the interference, while making little change to the target signal. The proposed method is compared with similar methods in both the anechoic and echoic cases. The method is shown to outperform the other methods in the anechoic case while in the echoic case it is shown to perform less well at reducing the level of the interference but still introduces the least artefacts. Extension to the proposed method to the N source and microphone case is also discussed.

1. INTRODUCTION

In a live sound performance it is common for multiple instruments or musicians to be performing at the same time. A common technique for setting microphones in this situation is to place a dedicated microphone to reproduce each sound source. Ideally, a given microphone signal will only contain the sound from a single source. In reality, a microphone may reproduce any number of sources surrounding it. This is similar to the concept of crosstalk in telecommunications and can be called bleed or leakage.

A microphone reproduces sound that enters the area surrounding it which is described by its polar pattern. When placing a microphone to reproduce a target sound source, it is placed to ensure the source is within this area. Sound from other sources may also enter this area and will also be reproduced, which can be referred to as interference. This interfering signal is assumed to consist of target signals of other microphones, as shown in Figure 1 and described in Equation (1)

$$\begin{aligned} x_1[n] &= \alpha_{11}s_1[n - \tau_{11}] + \alpha_{21}s_2[n - \tau_{21}] \\ x_2[n] &= \alpha_{12}s_1[n - \tau_{12}] + \alpha_{22}s_2[n - \tau_{22}], \end{aligned} \quad (1)$$

where x_1 and x_2 are microphone signals at timestep n , s_1 and s_2 are the sound sources, τ_{11} , τ_{12} , τ_{21} and τ_{22} are the delays of each source arriving at each microphone and α_{11} , α_{12} , α_{21} and α_{22} represent gain.

Interference of other sources causes a number of problems. An interfering signal can be a nuisance and can reduce the intelligibility of the target source. It can affect the overall gain of the microphone signal. It also means that if any processes are applied with

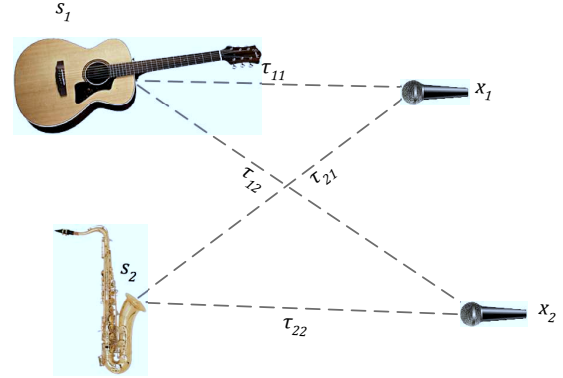


Figure 1: A configuration of 2 sources being reproduced by 2 microphones with the direct signal paths and equivalent delays shown.

the intention of being applied to the target signal, such as equalisation, it will also be applied to the interfering signal potentially causing problems in a mix.

Microphone interference can also cause comb filtering. Comb filtering occurs when a signal and a delayed version of the same signal are summed. A comb filter has defined peaks and troughs in the frequency response, caused by reinforcement and cancellation in the frequency domain. The comb filtering effect can be heard when the duplicated source is as much as 18dB quieter than the original [1].

If the microphone signals defined in Equation (1) are summed to the output y this becomes

$$y[n] = x_1[n] + x_2[n] \quad (2)$$

$$\begin{aligned} &= \alpha_{11}s_1[n - \tau_{11}] + \alpha_{12}s_1[n - \tau_{12}] + \\ &\quad \alpha_{21}s_2[n - \tau_{21}] + \alpha_{22}s_2[n - \tau_{22}] \end{aligned} \quad (3)$$

assuming

$$\tau_{11} < \tau_{21} \quad (4)$$

$$\tau_{22} < \tau_{12}. \quad (5)$$

Equation (3) shows that two versions of each source with different delays will be summed, thus causing comb filtering of both sources. The relative difference of the delay of each source arriving at each microphone is defined by

$$\tau_1 = \tau_{21} - \tau_{11} \quad (6)$$

$$\tau_2 = \tau_{12} - \tau_{22} \quad (7)$$

* The test audio in this research was excerpts of the raw audio of "Ana" by Vieux Farka Touré, available under a Creative Commons Attribution-NonCommercial license.

and the relative gain difference as

$$\alpha_1 = \alpha_{21} - \alpha_{11} \quad (8)$$

$$\alpha_2 = \alpha_{12} - \alpha_{22}. \quad (9)$$

Microphone and instrument placement plays an important role in the amount of microphone bleed that occurs. In a studio situation, for example, instruments can be isolated either in separate live rooms or by erecting baffles to provide some sound isolation. In a live sound situation this is not aesthetically appropriate. Microphone placement can be used to an advantage by using directional microphones and placing interfering signals in the null areas of a microphone's pick up area. This will not eliminate all interference and could cause other artefacts to occur, such as problems in the low frequencies due to the proximity effect.

1.1. Blind Source Separation

This problem can be looked at from a Blind Source Separation (BSS) point of view. BSS methods attempt to extract N sources from a mixture. The work in this paper is aimed at live sound where it is imperative that a method is able to run in real time. BSS methods generally are offline processes but a number of real-time implementations exist such as [2] and [3]. The method in [3] is taken from the Duet method of source separation, first presented in [4] and extended in [5]. Although stated to run in realtime, this method of source separation is aimed at the unmixing of N sources from 2 mixtures, i.e. from a stereo mix of panned sources. It is possible to use this method for 2 microphone recordings, but there are limitations to the distance between the microphones, which is reliant on sampling frequency for example at 16kHz the maximum distance allowed between the microphones for the method to run is when $d \leq 2.15\text{cm}$ [5]. The method in [2] is also used for stereo mixtures, assuming there is phase coherence between the mixtures and only intensity differences. This cannot be assumed in the multiple microphone case.

1.2. Noise Cancellation

Many of the problems that affect live sound are also present in telecommunications, for example noise and reverberation. Techniques exist in telecommunications for echo and noise cancellation, which share the same principles, and also run in real-time. The drawback is that most techniques are optimised for voice signals with lower bandwidths, for example a sampling rate of 8kHz is common [6] whereas in live sound we require a bandwidth to represent all the audible frequencies from 20Hz to 20kHz. For this reason, when an algorithm optimised for voice application is extended to incorporate wider bandwidth signals, the computational cost inherently increases.

In telecommunications, it is common that an external noise source will interfere with the direct source, for example a person speaking into a telephone may also have an interfering noise, such as air conditioning, in the same room. If an adequate estimation of the noise source is possible, this can be removed from the direct signal. This is where noise and echo cancellation can be used.

Common techniques for noise cancellation make use of an adaptive filter to estimate the impulse response of the interference of the noise signal to the main signal. These methods rely on a clean reference of the noise signal. In reality, this is not always the case. In a live sound scenario, a clean reference signal may not be

available as microphone bleed is assumed to be occurring on all signals.

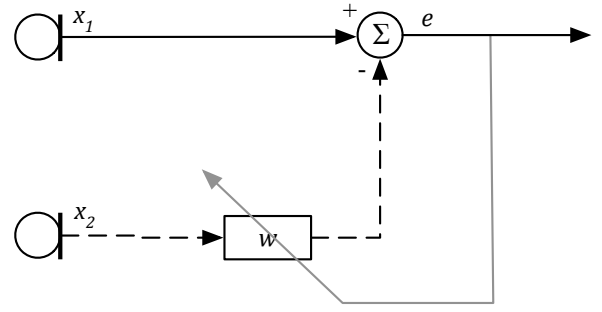


Figure 2: Block diagram of an adaptive filter

The Least Mean Squares (LMS) adaptive filter error is calculated by

$$e[n] = x_1[n] - \mathbf{W}^T[n]\mathbf{X}_2[n], \quad (10)$$

where the error e is also the clean output signal, where

$$\mathbf{X}_2[n] = [x_2[n], x_2[n-1], \dots, x_2[n-L+1]]^T \quad (11)$$

and the estimated update filter is

$$\mathbf{W}[n] = [w_0, w_1, \dots, w_{L-1}]^T. \quad (12)$$

The estimate of $\mathbf{W}[n]$ is then calculated by minimising $E\{e^2[n]\}$

$$\mathbf{W}[n+1] = \mathbf{W}[n] + \mu e[n]\mathbf{X}_2[n], \quad (13)$$

where μ is the adaptation step, which is generally a small value that effects convergence speed and accuracy, and the error signal e is the clean output.

Work in [7] addresses the same problem assuming close microphones and finding the Wiener filter solution by Power Spectral Density (PSD) estimation.

2. CTRANC

A crosstalk resistant adaptive noise canceller (CTRANC) [8] assumes there is crosstalk or as it is referred to in this paper, microphone bleed, between the microphones. For this reason a clean reference is not assumed. Adaptive filters are then cascaded so the output of one becomes the input of the other, as shown in Figure 3. In this way, once one signal has the interference cancelled out it can be used as the reference for the interference cancellation of another source and vice versa [9], [6]. The LMS algorithm then becomes

$$e_1[n] = x_1[n] - \mathbf{W}_A^T \mathbf{E}_2[n] \quad (14)$$

$$e_2[n] = x_2[n] - \mathbf{W}_B^T \mathbf{E}_1[n], \quad (15)$$

where the FIR adaptive filters are

$$\mathbf{W}_A[n] = [w_1^A[n], w_2^A[n], \dots, w_N^A[n]] \quad (16)$$

$$\mathbf{W}_B[n] = [w_1^B[n], w_2^B[n], \dots, w_N^B[n]] \quad (17)$$

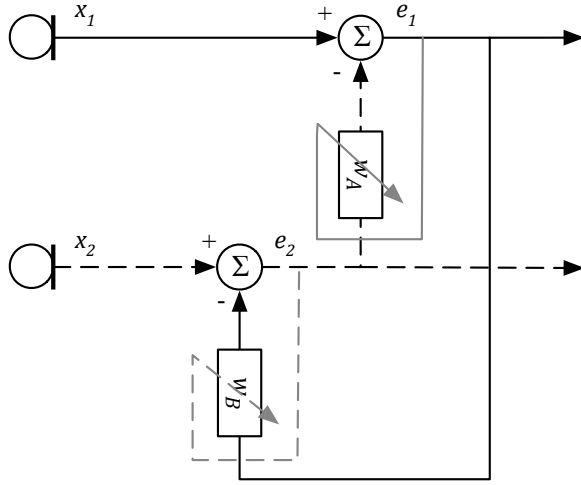


Figure 3: Block diagram of the CTRANC with 2 sources

and error vectors are

$$\mathbf{E}_1[n] = [e_1[n], e_1[n-1], \dots, e_1[n-N]] \quad (18)$$

$$\mathbf{E}_2[n] = [e_2[n], e_2[n-1], \dots, e_2[n-N]]. \quad (19)$$

Each filter is then updated by

$$\mathbf{W}_A[n+1] = \mathbf{W}_A[n] + \mu \mathbf{E}_2 e_1[n] \quad (20)$$

$$\mathbf{W}_B[n+1] = \mathbf{W}_B[n] + \mu \mathbf{E}_1 e_2[n]. \quad (21)$$

3. CENTRED ADAPTIVE FILTERS

In the purely anechoic case, the output of the adaptive filter in Equation (13) will simply be a single peak at a position representing delay and an amplitude representing gain and all other values are assumed to be 0. In reality, with the addition of reverberation and noise there will be a noise floor but there will still be a peak at the delay position. If the delay value is known, it is then possible to update fewer coefficients to get an accurate estimation of gain. Fewer coefficients means faster and more accurate convergence and less computational cost. Only a rough estimation of delay is required as a window of coefficients around the estimated delay value are updated. If the delay estimation is inaccurate by less than the window size then the method will still converge to the solution [10], [11], [12].

As in the LMS adaptive filter, the error is defined as

$$e[n] = x_1[n] - \mathbf{W}^T[n] \mathbf{X}_2[n] \quad (22)$$

and the filter coefficients updated using

$$\mathbf{W}[n+1] = \mathbf{W}[n] + \mu e[n] \mathbf{X}_2[n], \quad (23)$$

but where

$$\mathbf{W}[n] = [w_{\delta-D}[n], \dots, w_{\delta+D}[n]] \quad (24)$$

$$\mathbf{X}_2[n] = [x_2[n-\delta-D], \dots, x_2[n-\delta+D]], \quad (25)$$

and where δ is the estimation of the delay and D is a user-defined error distance around the delay to update the coefficients. A higher value of D will yield slower convergence but will encompass additional echoes or reverberation.

3.1. Delay estimation

There are many delay estimation methods [13] for estimating δ , equivalent to τ_1 and τ_2 in Equations (6) and (7). Adaptive filters themselves can be used to estimate delays [14], but this has the same computational cost that is trying to be avoided. A common method used is the Generalized Cross Correlation (GCC), first introduced in [15]. This method is computationally cheap and allows weightings to be applied to improve performance against noise and reverberation, such as the Phase Transform (PHAT).

The GCC is calculated using

$$\Psi = \mathcal{F}^{-1} \{X_1^*[k] \cdot X_2[k]\}, \quad (26)$$

where Ψ is the GCC, \mathcal{F}^{-1} denotes the Inverse Fast Fourier Transform, $*$ denotes the complex conjugate and X_1 and X_2 are x_1 and x_2 in the frequency domain. By applying the PHAT, Ψ becomes

$$\Psi_P = \mathcal{F}^{-1} \left\{ \frac{X_1^*[k] \cdot X_2[k]}{|X_1[k] \cdot X_2[k]|} \right\}, \quad (27)$$

where $|\cdot|$ denotes the absolute magnitude. The estimate of delay δ is then calculated by

$$\delta = \arg \max_n \Psi_P[n]. \quad (28)$$

To reduce computational cost, it is also possible to calculate multiple delays from a single GCC-PHAT calculation [16] by extracting the position of N peaks rather than just 1.

4. CENTRED CTRANC

This paper proposes combining the CTRANC with the centred adaptive filters, known as centred CTRANC, to improve performance and convergence of the CTRANC method. As the CTRANC method the error signals are defined as

$$e_1[n] = d_1[n] - \mathbf{W}_A^T \mathbf{E}_2[n] \quad (29)$$

$$e_2[n] = d_2[n] - \mathbf{W}_B^T \mathbf{E}_1[n], \quad (30)$$

but where

$$\mathbf{W}_A[n] = [w_{\delta_1-D}^A[n], \dots, w_{\delta_1+D}^A[n]] \quad (31)$$

$$\mathbf{W}_B[n] = [w_{\delta_2-D}^B[n], \dots, w_{\delta_2+D}^B[n]] \quad (32)$$

and

$$\mathbf{E}_1[n] = [e_1[n-\delta_1-D], \dots, e_1[n-\delta_1+D]] \quad (33)$$

$$\mathbf{E}_2[n] = [e_2[n-\delta_2-D], \dots, e_2[n-\delta_2+D]] \quad (34)$$

and the filter coefficients are updated using

$$\mathbf{W}_A[n+1] = \mathbf{W}_A[n] + \mu \mathbf{E}_2 e_1[n] \quad (35)$$

$$\mathbf{W}_B[n+1] = \mathbf{W}_B[n] + \mu \mathbf{E}_1 e_2[n], \quad (36)$$

which requires estimation of both δ_1 and δ_2 .

5. DIRECT CALCULATION

The aim of using adaptive filters is to estimate the delay and gain changes of the sources arriving at each microphone, as defined by Equations (6) - (9). It is therefore possible to directly calculate the delay and gain difference for each frame of audio. This information is then used to scale and delay the interference, which is then subtracted from the direct signal, thus removing the interference.

This method is not commonly used as it requires averaging to simply provide a stable solution. For example the amplitude difference calculated for each frame will be slightly different, thus causing amplitude modulation of the interference. The adaptive filters have to converge to a solution which is then stable and will not modulate the signal. This method also does not take into account the crosstalk, relying on a clean interference signal.

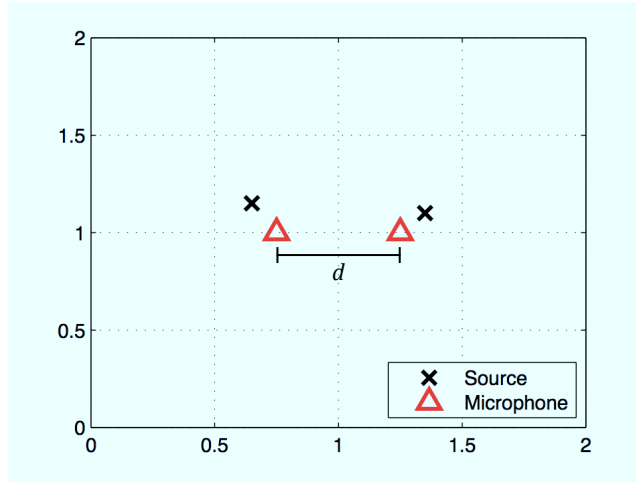


Figure 4: Simulation microphone and source layout where $d = 0.5\text{m}$

6. COMPARISON

The methods outlined in this paper were compared by measuring the amount of interference reduction and how artifacts and distortion effect the target sound. The CTRANC and centred CTRANC methods were optimised to produce the best results by selecting a suitable value for the adaption step, μ and the error distance D . The methods were compared in the 2 source, 2 microphone case.

6.1. Simulation Experimentation

The methods were first compared using simulated microphone signals. The sources and microphones were positioned virtually and the equivalent delay and gain calculated for each source to each microphone. The input sources were a guitar and vocal track. The sources were then combined to simulate each microphone signal with bleed. The microphones were assumed to be omnidirectional in an anechoic environment. The sources were placed between 10cm and 12cm from the microphones, as shown in Figure 4. The distance d was increased from 10cm to 5m, producing different values for delay and gain. The relative position of each source to each microphone remained the same.

6.2. Results

The simulated microphone outputs containing the direct source and lower amplitude bleed were then passed through each method. The results were analysed using the BSS_EVAL Matlab toolbox [17] to extract the signal-to-interference (SIR_{dB}), signal-to-artefact (SAR_{dB}) and signal-to-distortion (SDR_{dB}) ratios. The unprocessed microphone signals were also analysed for comparison. The results in this paper show the scenario where s_1 is the target signal and s_2 is the interfering signal.

Figure 5 shows the calculated SIR_{dB} for each method at each microphone distance of d . The centred CTRANC can be shown to have the highest values of SIR_{dB} for all but the $d = 0.1$ case, where DUET outperforms it. It is expected that the DUET method may perform well for small values of d as, although it is not aimed at microphone signals, it can perform source separation at small distances. The SIR_{dB} determines how much the interference has been reduced. As mentioned previously, studies have shown that comb filtering can be heard when the duplicate source is as much as 18dB lower in amplitude than the original [1]. It can be seen that the centred CTRANC reduces the level of the interference by more than 18dB for each value of d , therefore the possibility of comb filtering will be removed, even if the interference is not completely cancelled out.

The Wiener filter method [7] proved to outperform the proposed method in certain instances of d for SIR_{dB} but overall performed inconsistently over all values of d in the simulation experiment. The Wiener filter method assumes each microphone is an approximation of the ideal impulse response of the direct sound path and that the interference is of a lower amplitude. If the interference is of a high enough amplitude, this assumption will no longer hold.

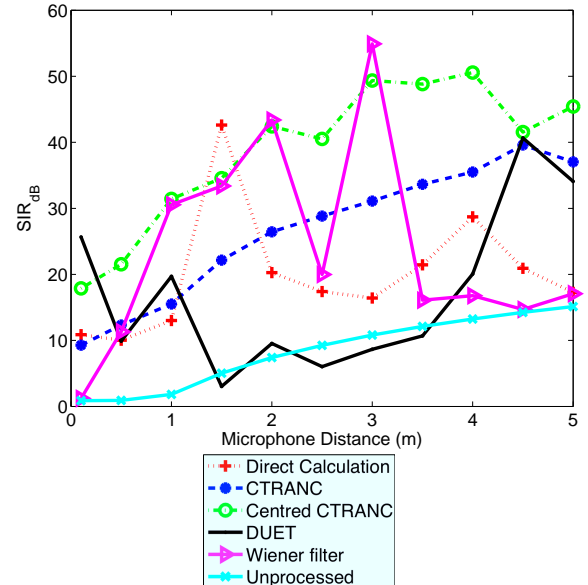


Figure 5: Signal-to-interference ratio of each method at each iteration of microphone distance for the simulated case.

Although DUET performs best on SIR_{dB} at $d = 0.1$ Figure 6 shows the centred CTRANC has a higher value of SAR_{dB} at the same distance. Signal to artefact ratio describes the amount of arte-

facts that have been introduced by a method. This shows that the DUET method introduces a lot of artefacts to the processed signal. Methods based on adaptive filters will generally not add additional artefacts to the target source as it is attempting to subtract the interfering source in the time domain. In live sound, this is desired as it would be preferable to remove some of the interference but leave the target signal intact rather than completely remove the interference but heavily distort the target signal. The results shown in Figure 7 also agree with this.

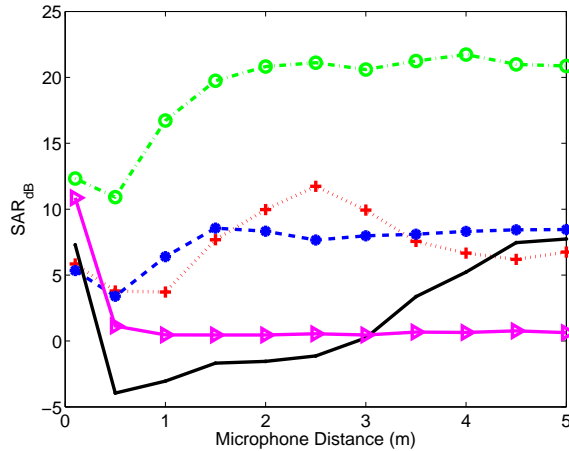


Figure 6: Signal-to-artefact ratio of each method at each iteration of microphone distance for the simulated case.

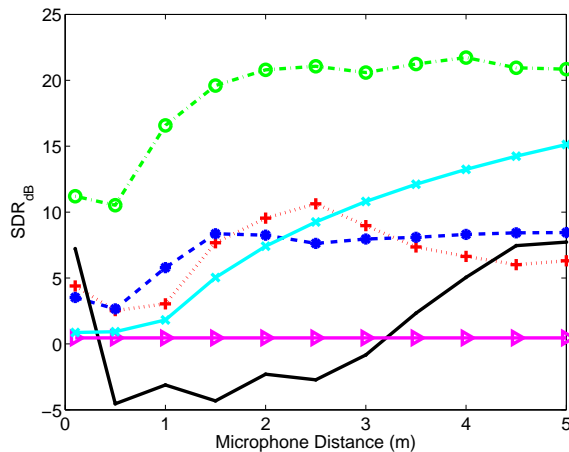


Figure 7: Signal-to-distortion ratio of each method at each iteration of microphone distance for the simulated case.

6.3. Real Recordings

To test each method's effectiveness in a real space, a test was setup using 2 speakers and 2 microphones. The speakers were spaced from 10cm to 100cm at 10cm intervals while the microphones were always placed 10cm from each speaker, with an error of ± 1 cm as in Figure 8. This distance was chosen to simulate a close microphone configuration. It is not assumed the layout is symmetric.

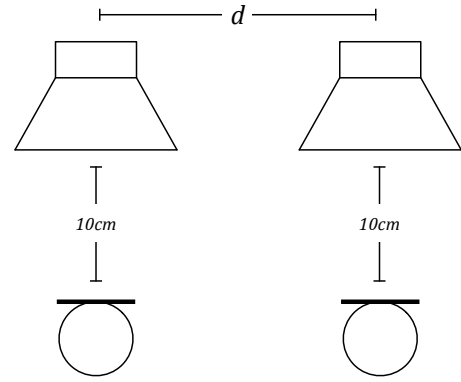


Figure 8: Layout of speakers and microphones in the test recordings

6.4. Results

As with the simulation, the SIR_{dB} , SAD_{dB} and SDR_{dB} for each method and value of d was calculated and can be seen in Figures 9, 10 and 11. Figure 9 shows the method with the highest performance is the original CTRANC method. The reason for this is that due to the excess noise and reverberation, the centred CTRANC would produce errors in the impulse response at the edges of the window but would estimate the amplitude and delay so would improve the SIR_{dB} , as it has higher SIR_{dB} than the unprocessed microphone signals. Using a higher value of D may improve this, but by increasing D the computational cost increases. The Wiener filter method performed only slightly lower than the traditional CTRANC method. Unlike in the simulation experiments, the Wiener filter method performs more consistently with real recordings. The DUET method proved to be more successful at some lengths of d but is not consistent over all the distances tested.

Figures 10 and 11 show the SAR_{dB} and SDR_{dB} for the real test. As shown in the simulations, the DUET method adds additional delay and artefacts. The centred CTRANC overall performs best when measuring SAR_{dB} , agreeing with the simulations that the centred CTRANC does not add artefacts or distortion and is consistent over all values tested of d . In the real recordings the CTRANC has shown to perform consistently well, particularly by the SIR_{dB} measure, but performs worse than the centred CTRANC in measures of SAR_{dB} and SDR_{dB} , thus using the centred CTRANC also reduces the amount of artefacts and distortion over the CTRANC. The Wiener filter method performed worse than the centred CTRANC method but with slightly higher values of SAR_{dB} and SDR_{dB} than the traditional CTRANC. It can therefore be said that for SAR_{dB} and SDR_{dB} these methods had similar performance.

7. CONCLUSIONS AND FUTURE WORK

A centred CTRANC method has been proposed that combines centred adaptive filters with the CTRANC system of noise cancellation. The proposed method outperformed other methods for interference reduction in the simulated anechoic case with little additional artefacts compared to the other methods under test. The

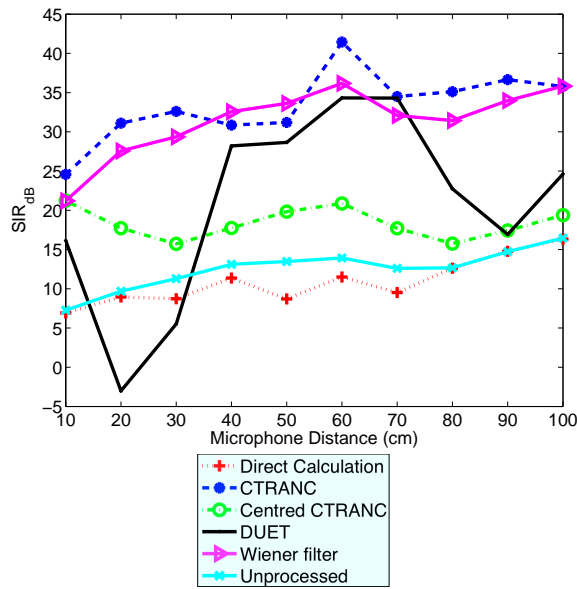


Figure 9: Signal-to-interference ratio of each method at each iteration of microphone distance for the real case.

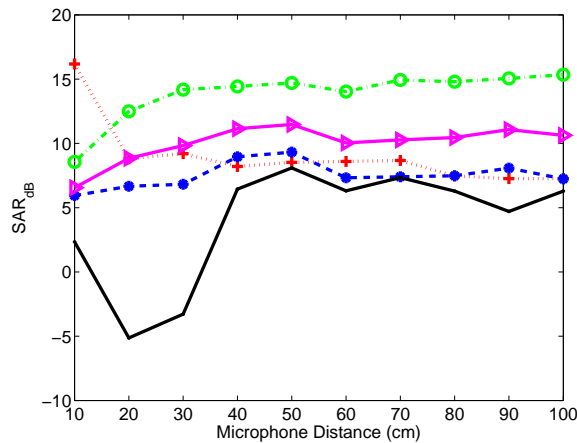


Figure 10: Signal-to-artefact ratio of each method at each iteration of microphone distance for the real case.

proposed method was shown to be outperformed by the simple CTRANC system in real recordings but was shown to also introduce less artefacts.

The efficacy of the centred CTRANC is not effected by the level of the interference but by the environment within which the sources and microphones are placed and the reverberation and noise present therefore it is currently best suited to close microphone applications. Work continues into modifying the centred CTRANC to improve the robustness in real environments, such as manipulating the filter output at each iteration using the estimate of delay by, for example, applying a weighting to the coefficient update range, assuming the correct delay lies close to the centre of the range. The time domain filter can also be assumed to only have positive coefficients therefore the negative components can be set to 0. The problem with this approach is it may effect the convergence prop-

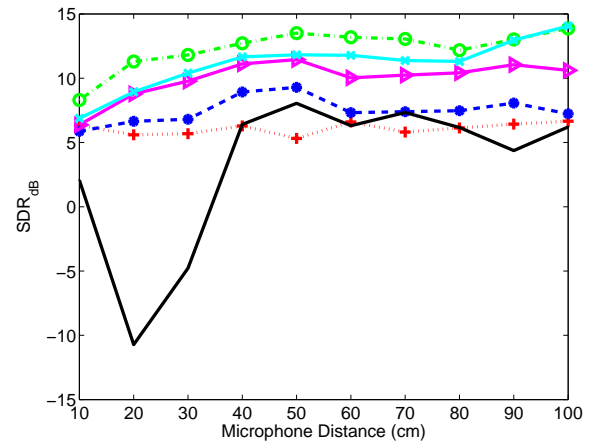


Figure 11: Signal-to-distortion ratio of each method at each iteration of microphone distance for the real case.

erties as the error is now no longer the calculated error.

7.1. CTRANC - N source and N microphones

Theoretically, the CTRANC method can be scaled to N sources and microphones, assuming the number of sources is equal to the number of microphones and therefore the system is homogenous. A diagram of this can be seen in Figure 12. As the number of source and microphones increases, the number of adaptive filters F required increases, with the relation $F = N(N - 1)$. For this reason computational complexity increases as N increases. Work will continue looking at whether centred filters can be implemented in the N source case to improve bleed reduction.

8. REFERENCES

- [1] S. Brunner, H.-J. Maempel, and S. Weinzierl, "On the audibility of comb-filter distortions," in *Proceedings of the 122nd Audio Engineering Society Convention*, (Vienna, Austria), 2007.
- [2] D. Barry, E. Coyle, and B. Lawlor, "Real-time sound source separation: Azimuth discrimination and resynthesis," in *Audio Engineering Society Convention 117*, 10 2004.
- [3] M. Baeck and U. Zölzer, "Real-time implementation of a source separation algorithm," in *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, (London, UK), 2003.
- [4] A. Jourjine, S. Rickard, and Ö. Yilmaz, "Blind separation of disjoint orthogonal signals: Demixing n sources from 2 mixtures," in *ICASSP '00: Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference*, vol. 5, (Istanbul, Turkey), 2000.
- [5] Ö. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Transactions on Signal Processing*, vol. 52, July 2004.
- [6] G. Mirchandani, J. Zinser, R.L., and J. Evans, "A new adaptive noise cancellation scheme in the presence of crosstalk

- [speech signals],” *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 39, pp. 681–694, Oct. 1992.
- [7] E. K. Kokkinis and J. Mourjopoulos, “Unmixing acoustic sources in real reverberant environments for close-microphone applications,” *J. Audio Eng. Soc.*, vol. 58, no. 11, pp. 907–922, 2010.
 - [8] J. Zinser, R., G. Mirchandani, and J. Evans, “Some experimental and theoretical results using a new adaptive filter structure for noise cancellation in the presence of crosstalk,” in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, vol. 10, pp. 1253 – 1256, Apr. 1985.
 - [9] L. Lepauloux, P. Scarlart, and C. Marro, “An efficient low-complexity algorithm for crosstalk-resistant adaptive noise canceller,” in *17th European Signal Processing Conference (EUSIPCO 2009)*, 2009.
 - [10] V. Margo, D. Etter, N. Carlson, and J. Gross, “Multiple short-length adaptive filters for time-varying echo cancellations,” in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 1, pp. 161–164 vol.1, Apr. 1993.
 - [11] Y. Lu, R. Fowler, W. Tian, and L. Thompson, “Enhancing echo cancellation via estimation of delay,” *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4159–4168, 2005.
 - [12] J. Gross, D. Etter, V. Margo, and N. Carlson, “A block selection adaptive delay filter algorithm for echo cancellation,” in *Circuits and Systems, 1992., Proceedings of the 35th Midwest Symposium on*, pp. 895–898 vol.2, Aug. 1992.
 - [13] J. Chen, J. Benesty, and Y. A. Huang, “Time delay estimation in room acoustic environments: An overview,” *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. 1 – 19, 2006.
 - [14] F. Reed, P. Feintuch, and N. Bershard, “Time delay estimation using the lms adaptive filter - static behaviour,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 3, p. 561, 1981.
 - [15] C. H. Knapp and G. C. Carter, “Generalized correlation method for estimation of time delay,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 24, no. 4, pp. 320–327, 1976.
 - [16] A. Clifford and J. Reiss, “Calculating time delays of multiple active sources in live sound,” in *Proceedings of the 129th Audio Engineering Society Convention*, November 2010.
 - [17] C. Févotte, R. Gribonval, and E. Vincent, “Bss eval toolbox user guide,” tech. rep., IRISA Technical Report 1706, Rennes, France, April 2005.

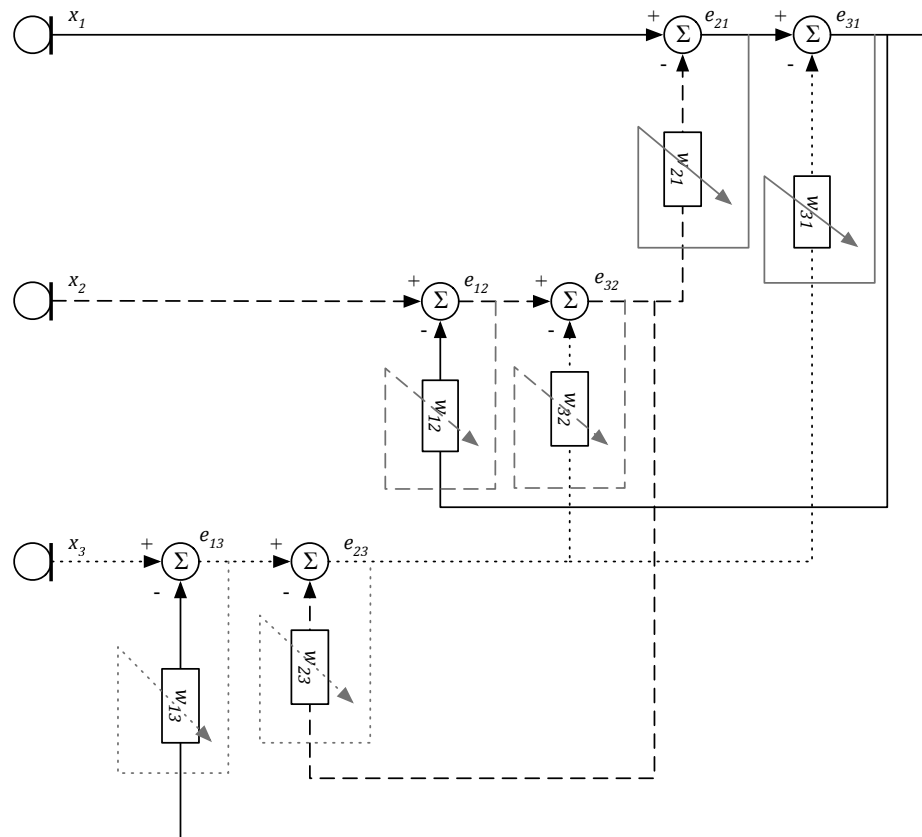


Figure 12: Block diagram of the CTRANC method with 3 microphone inputs and 3 sources.

THE IMAGE-SOURCE REVERBERATION MODEL IN AN N-DIMENSIONAL SPACE

Stephen McGovern

Spacenet, Inc.

McLean, VA, United States

smcgover@alumni.stevens.edu

ABSTRACT

The image method is generalized to geometries with an arbitrary number of spatial dimensions. n -dimensional (n -D) acoustics is discussed, and an algorithm for n -D room impulse response calculations is presented. Synthesized room impulse responses (RIRs) from n -D rooms are presented. RIR characteristics are discussed, and computational considerations are examined.

1. INTRODUCTION

Reverberation is a widely used and indispensable audio effect. It is used to simulate environmental acoustics and as a means for artistic modification of audio.

Descriptions of reverberation are often broken into two portions, the early echoes and the tail. The characteristics of each portion is important for predicting the resultant effect [1] [2] [3]. Early echoes are important for conveying spatial information to the listener. This is often implemented in commercial reverberators as a tunable parameter referred to as "pre-delay." The tail portion of the reverberation response can vary greatly in length and thereby determines how long reverberation persists after an excitatory input.

Before digital electronics became commonplace, room acoustic simulation was achieved through analog means. Audio signals were processed mechanically by transmitting vibrations through a spring or a plate. Today, digital emulations of springs and plates are popular with musicians and they continue to be the topic of many papers such as [4] [5] and [6] [7], respectively.

Comb and all-pass filters have long been used in reverberators. The approach can be generalized as a feedback delay network (FDN) [8] and it has the benefit of having relatively low computational overhead [9].

Various wave-based models have been a popular research topic. They have the drawback of being computationally intensive, although progress is being made to accelerate computation [10].

It is also possible to measure an RIR using a real room. This may be done either for analysis or as an finite impulse response (FIR) filter for direct simulation. Implementation of FIR filters tend to require greater overhead than FDNs; however, this is now less of an issue as computers have become more powerful. Methods such as segmented convolution can be used to reduce latency [9].

RIRs can also be synthesized using methods based in geometrical acoustics theory. Methods using this approach include mirror images [11], ray tracing [12], and, more recently, beam tracing [13]. Aside from the typical time domain models, frequency domain methods exist as well [14] [15]. The validity of these types of simulations was explored in [16].

While geometrically complex rooms have been simulated, the additional physical complexity requires higher algorithmic complexity. Models of box-shaped rooms, however, are sufficient for many purposes and they continue to be an active topic in research [9] [11] [14] [15].

There are other effects related to reverberation, though some might not clearly meet the criteria for what is commonly understood to be reverberation. These include delay, gated reverb, musical instrument body models [17] [18], and resonance effects [19]. While this paper will not deal with such concepts directly, it has been written in the context of the full scope of reverberation-related effects.

1.1. Review of Dimensionality in Existing Models

There has been very little research in the area of n -dimensional reverberation. There are, however, a few notable exceptions [20] [21] [22]. These papers discussed n -D DWM, hyperdimensional finite difference time domain (FDTD) mesh, and hyperdimensional DWM, respectively.

In contrast, most reverberation algorithms are either developed in reference to some type of 3-D acoustic space or they seek to directly model a 3-D acoustic space. Examples of this include wave-based methods [10], the image method [23], and beam tracing [13].

Aside from 3-D models, equivalent 2-D models have been studied as well. Two dimensional digital waveguide mesh (DWM) implementations have been used to model both 2-D acoustic spaces [24] [25] and vibrations on plates and membranes [26]. A 2-D image method has been used to model room reverberation [27], and a 2-D beam tracer has been used to model acoustics in [28] [29].

Outside of the field of room simulation, the image method has been used to model vibration in 1-D beams [30], 1-D beams and 2-D plates [31], and 2-D plates [32]. While there is little published discussion of reverberation models based on 1-D systems, such models produce delay effects. Interestingly enough, delay and reverberation are easily confused by novice musicians.

What is lacking in all of this is a unified model characterizing 1-D, 2-D, and 3-D systems using a method based on geometrical wave theory. There is also no such model comparable to the mesh simulations described in [21] [22] that is extendable to higher dimensions. There is very little formal discussion on why 1-D, 2-D, and 3-D models all produce similar effects, and there is a void of research regarding how higher dimensional resonant systems ought to sound.

Sec. 2 discusses background mathematics related to RIR calculations. Sec. 3 examines n -D geometry and its implications for rooms and acoustics. Algorithmic implementation is discussed in Sec. 4. In Sec. 5, implications of n -D room simulations are

analyzed. An approach for modeling non-integer dimensions is presented in Sec. 6. Sec. 7 examines algorithmic efficiency, presents benchmarks, and then discusses simulations. Finally, Sec. 8 presents the conclusions.

2. BACKGROUND MATHEMATICS

The image method generates an impulse response by creating mirror images of a sound source across the walls of an enclosure. The acoustical pressure impulse of the image with index q is given by

$$p(r, t)_q = \frac{A_q}{|\mathbf{r} - \mathbf{r}_q|} \delta \left(t - \frac{|\mathbf{r} - \mathbf{r}_q|}{c} \right) \quad (1)$$

where A_q is a constant, $|\mathbf{r} - \mathbf{r}_q|$ is the distance between the receiver and the q th source, δ is the Dirac delta function, and c is the speed of sound. The pressure impulses are then summed together as if they are a superposition of independent sources. This results in the pressure impulse response function given by

$$h(t) = \sum_q p(t)_q \quad (2)$$

where the summation takes place over all images in 3-D.

By taking the Fourier transform of Eq. 1 the acoustical pressure impulse may be given in the frequency domain as

$$p(w, t)_q = \frac{A_q}{|\mathbf{r} - \mathbf{r}_q|} e^{-j \frac{w}{c} |\mathbf{r} - \mathbf{r}_q|} \quad (3)$$

This results because of a property of the Dirac delta function. This property was given in [33]. In a fashion similar to Eq. 2, the summation may be taken in the frequency domain. This results in

$$H(w) = \sum_q p(w, t)_q \quad (4)$$

In [34] it was shown that the inverse Fourier transform of Eq. 4 could be taken to yield an expression equivalent to Eq. 2.

$$h(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} H(w) e^{j\omega t} d\omega \quad (5)$$

n -D box-shaped enclosures can preserve much of the simplicity of 3-D box-shaped enclosures, while also permitting impulse responses with more greatly varying properties.

By inverting the derivation in the appendix of [23], a modal frequency solution for the rigid walled rectangular-room boundary value problem can be obtained from Eq. 2. In [21], this solution was given for an n -D room in the form

$$p_{d_1 d_2 \dots d_n}(x_1, x_2, \dots, x_n) = C \prod_{i=1}^n \cos \left(\frac{d_i \pi x_i}{L_i} \right) \quad (6)$$

where d_i is an integer, n is the total number of spatial dimensions, L_i is the length of the room along the i th dimension, and C is a constant. What follows from here will be a time-domain solution to the n -D normal mode expansion using the image method.

3. N-DIMENSIONAL SPACE

3.1. n -D Acoustic Space

A room in n -D occupies a volume with units (distance) n . It is bound by a hypersurface with units (distance) $^{n-1}$. The concept of 2-D walls is based on the presumption of a 3-D space. The concept of 2-D walls is thus replaced with the concept of $(n-1)$ -D reflectors. As for time, it will be considered distinct from space and it will not be treated as a dimension. The discussion here will be limited to rectilinear room shapes, as other shapes in n -D space are beyond the scope of this paper.

Any dimension of space has both a positive and a negative direction. For an n -D room, the source and receiver, along any one dimension, are located between two reflectors. The total number of reflectors in an n -D room is thus equal to $2n$. A room in 4-D is depicted in Fig. 1.

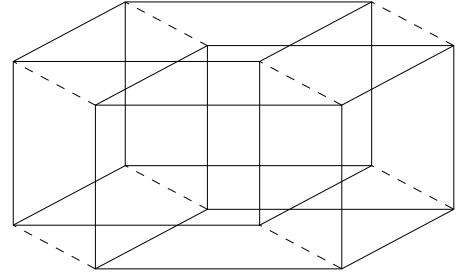


Figure 1: A 2-D parallel projection of a 4-D rectangular room.

3.1.1. Real Physical Systems for 1-D and 2-D

There are few, if any, tangible real-world systems analogous to an acoustic space with more than three dimensions. Acoustics tubes and plates, however, behave in a fashion similar to 1-D spaces and 2-D spaces, respectively. Waves in tubes are truly 1-D propagation of acoustic waves. Waves on plates propagate in 2-D, however, the propagation medium is metal and the waves are transverse rather than longitudinal. In [6] a comparative analysis was made between plates and rooms. Among the major differences is a phenomenon known as frequency dispersion. Frequency dispersion causes the propagation speed of a wave to vary with frequency. This occurs very significantly in plates, but not in rooms.

3.2. Reflection Coefficients

The constant A_q in Eqs. 1 and 3 includes a factor for the total reflection coefficient, which is itself the product of a set of reflection coefficients. Each reflection coefficient is given as $\beta = \pm \sqrt{1 - \alpha}$, where α is the absorption coefficient [34]. Classically the value of β is a positive number on the interval $[0, 1]$. Negative coefficients, however, have been studied as well [34] [35] [36]. In contrast to 3-D rooms, the reflection coefficients on 2-D plates are classically negative [31] [32]. This results in a reversal of the wave phase each time it is reflected by the plate boundary.

The value of a reflection coefficient depends on a number of factors. Because the existence of a higher dimensional room would require exotic forms of matter, the possible values for reflection coefficients can reasonably be presumed to include the interval $[-1, 1]$.

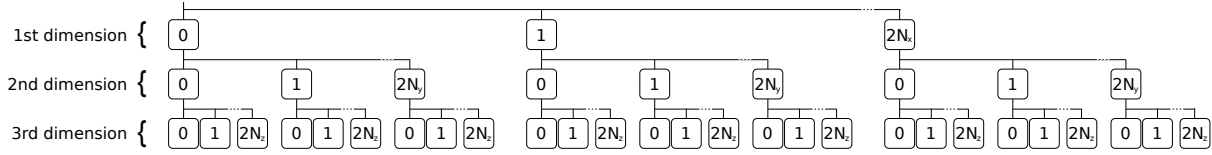


Figure 2: Iteration tree for calculation in 3-D. Dimension number increases top to bottom. Look-up table indices increase left to right. Image points are calculated for each leaf starting with the leftmost and moving to the right.

3.3. Energy Conservation and Pressure Amplitude

A sound source emits a finite amount of power. It is known that the distance attenuation of acoustic intensity is proportional to $1/r$ and $1/r^2$, for 2-D and 3-D, respectively [37]. Specifically in 3-D, acoustical intensity, or the power flux density, is $I_3 = P/4\pi r^2$, where the denominator is the area of a sphere with radius r . In 2-D, or equivalently, a cylindrical wave, the intensity is $I_2 = P/2\pi r$. Similarly, the intensity in a 1-D acoustic tube is $I_1 = P/2r^0$.

Integrating I_n for a source with power P over a closed Gaussian surface must yield the same value regardless of the problem's dimensionality. This results from the law of conservation of energy, and is shown as

$$\oint I(P)_m dS_m = \oint I(P)_n dS_n \quad (7)$$

where n and m are the number of dimensions. The expression for n -D power flux density can be written $I_n = P/S_{sn}$, where S_{sn} is the surface area of the n -D sphere. For a mention of several formulas for n -D geometry see Appendix A.

The calculation of an RIR requires a formula that relates sound pressure amplitude with distance. This formula should also be consistent with Eq. 7 and, hence, obey the law of energy conservation. Such a formula is given by

$$p(r)_n = \frac{A}{\sqrt{r^{n-1}}} \quad (8)$$

A derivation of Eq. 8 is provided in Appendix B.

4. COMPUTATIONAL PROCEDURE

For boxed-shaped rooms, the summation can be taken over each dimension. In [11], an algorithm using the formula

$$h(t) = \sum_{i=0}^{2N_x} \sum_{j=0}^{2N_y} \sum_{k=0}^{2N_z} p(t)_{ijk} \quad (9)$$

was used where $p(t)_{ijk}$ is the pressure impulse resulting from the ijk^{th} image in 3-D. In this case, i , j , and k are indices for look-up tables rather than indices for an image lattice. The summation operation is depicted graphically in Fig. 2 as a tree diagram. The algorithm has a reduced number of floating point operations compared to prior algorithms and thus has better performance. For this reason, a modified version of the algorithm was chosen for n -D calculations. The primary modification to the algorithm was to the summation procedure. For n -D, Eq. 9 can be written as

$$h(t) = \sum_{i_1=0}^{2N_1} \sum_{i_2=0}^{2N_2} \sum_{i_3=0}^{2N_3} \dots \sum_{i_n=0}^{2N_n} p(t)_{i_1 i_2 i_3 \dots i_n} \quad (10)$$

The tree diagram in Fig. 2 has a fractal structure that can similarly be extended to any number of dimensions. To perform calculations for a truly arbitrary number of dimensions, a scheme that recursively stepped into higher dimensions was used.

Square of the distance summations and reflection coefficient product operations were collected recursively on each function call. This resulted in fewer floating point operations than [11]. To account for distance losses, a relation based on Eq. 8 was used. RIRs for several different dimensions are plotted in Fig. 3.

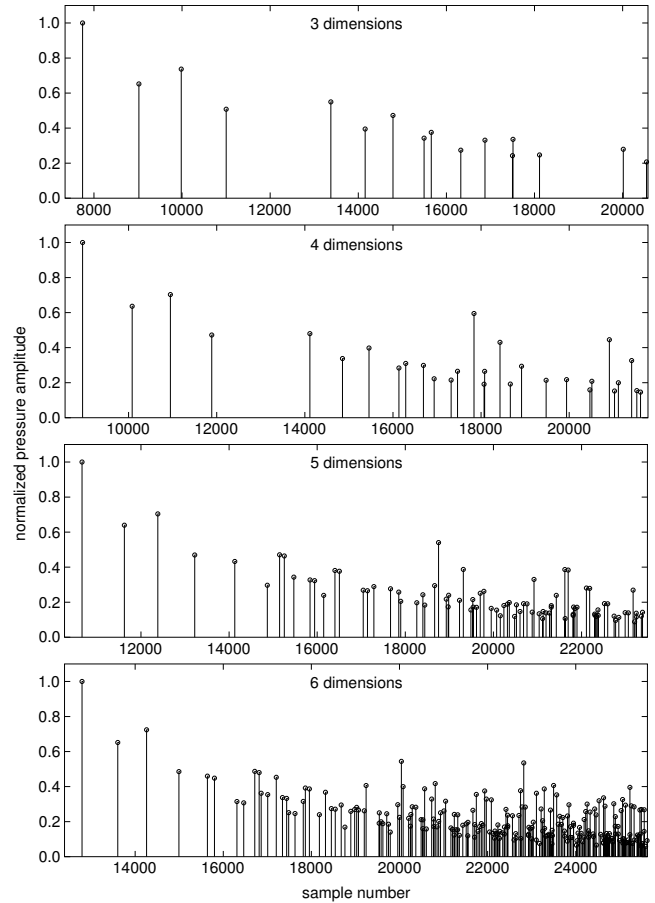


Figure 3: RIRs calculated for 3-D, 4-D, 5-D, and 6-D. Rooms measure approximately 10 meters in each dimension. RIRs are normalized and aligned to the first non-zero impulse.

5. ANALYTICAL IMPLICATIONS

The total number of echoes accounted for in an impulse response calculation of length t_{length} is given by the volume of an n -sphere of radius ct_{length} divided by the room volume. The result of this is given in by

$$E(t)_n = \frac{\pi^{\frac{n}{2}} c^n t^n}{\Gamma(\frac{n}{2} + 1) V_{rn}} \quad (11)$$

where Γ is the gamma function. The variable V_{rn} is the volume of the n -D room. The temporal echo density can be given by the time derivative of the total echo count, which leads to

$$\frac{dE}{dt}_n = \frac{2\pi^{\frac{n}{2}} c^n t^{n-1}}{\Gamma(\frac{n}{2}) V_{rn}} \quad (12)$$

On a 2-D plate, the echo density increases with t [6]. In a 3-D room, the echo density increases with t^2 . As is shown in Eq. 12, the echo density in an n -D space continues on with this pattern and increases with t^{n-1} . Plots for both Eqs. 11 and 12 are shown in Fig. 4.

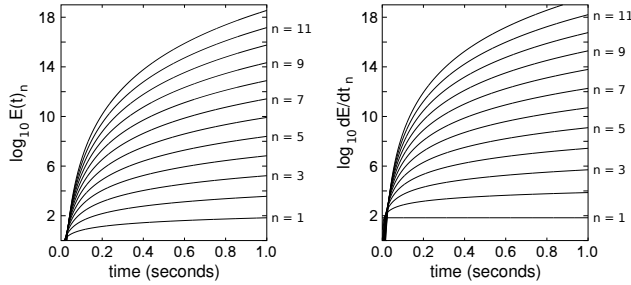


Figure 4: Plots for one to twelve spatial dimensions. left) total number of echoes, right) temporal echo density.

The literature describes various methods for calculating mixing time, many using different definitions [38] [39]. Generally, mixing time can be described as the transition from the early part of the response to the late part of the response. A commonly used definition for the point of transition is when the temporal echo density reaches some particular value. According to [2], this density is around 2000 - 4000 echoes / sec. Using 3000 echoes / sec as the transition point, the mixing time, t_{mix} , in n -D can be estimated as

$$t_{mix} = \left(3000 \frac{\Gamma(\frac{n}{2}) V_{rn}}{2\pi^{\frac{n}{2}} c^n} \right)^{\frac{1}{n-1}} \quad (13)$$

The mixing time is plotted for 1 to 45 dimensions in Fig. 5 on the left. It may be noted that a minimum occurs near $n = 8$. This occurs because dE/dt_n intersects dE/dt_m for $n \neq m$. This intersection is shown on the right in Fig. 5.

A curious effect that results from introducing additional dimensions to a given room configuration is that it increases the delay to existing impulses. Mathematically, this can be shown with an n -D extension of the Pythagorean theorem. It is illustrated in Fig. 3 where the delay time of the first impulse increases as the number of dimensions is increased. Here, the delay is observed as a shifted sample-time index. The delay can also be seen later in Fig. 6 as a change in distance to the nearest image point.

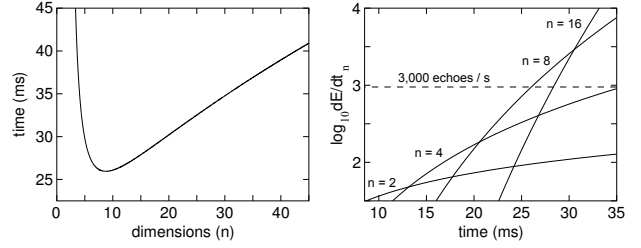


Figure 5: left) mixing times for rooms of n dimensions with $V_{rn} = 8^n$, right) temporal echo density for 2-D, 4-D, 8-D, and 16-D.

Because the computational costs have the potential to increase dramatically for higher dimensions, care must be taken in choosing input parameters. Table 1a shows, for a given dimension, the increase in the number of images as a factor when the time-length of the output RIR is increased by a factor of 2.

Table 1: a) Increase in sphere volume when the time-length of the impulse response is increased by a factor of 2. b) Required room volume V_{rm} when $E(t)_m = E(t)_n$ and $r = 68.6$.

(a)		(b)	
n	$V(2t)_{sn}/V(t)_{sn}$	m	V_{rm}/V_{r3}
1	2	1	1.01 e -04
2	4	2	1.09 e -02
3	8	3	1.00 e +00
4	16	4	8.08 e +01
5	32	5	5.91 e +03
6	64	6	3.98 e +05
7	128	7	2.50 e +07
8	256	8	1.47 e +09
9	512	9	8.21 e +10
10	1024	10	4.35 e +12

The higher computational costs of higher dimensional RIRs can be offset by using rooms with larger volumes. This is true presuming that the total time-length of the output RIR is held fixed. Computational costs for higher dimensional rooms will be similar to that of lower dimensional rooms if the total number of image points considered in each calculation is also similar. To estimate an m -D room volume that will produce an RIR with a similar number of image points as a given n -D room, the number of echoes from Eq. 11 can be set so that $E(t)_m = E(t)_n$. Solving for V_{rm} results in

$$V_{rm} = \frac{\pi^{\frac{m}{2}} r^m}{\pi^{\frac{n}{2}} r^n} \frac{\Gamma(\frac{n}{2} + 1)}{\Gamma(\frac{m}{2} + 1)} V_{rn} \quad (14)$$

The relative increase in volume resulting from Eq. 14 is shown in Table 1b for $n = 3$, $r = 68.6$ and various values of m .

If the total number of images can not be reduced, there are still other approaches that may prove useful for modeling n -D systems. The tail portion of RIRs have a strong noise-like behavior. Modeling this behavior has been shown to reduce computational costs for 3-D systems [40] and it is likely that this type of approach can also be extended to n -D.

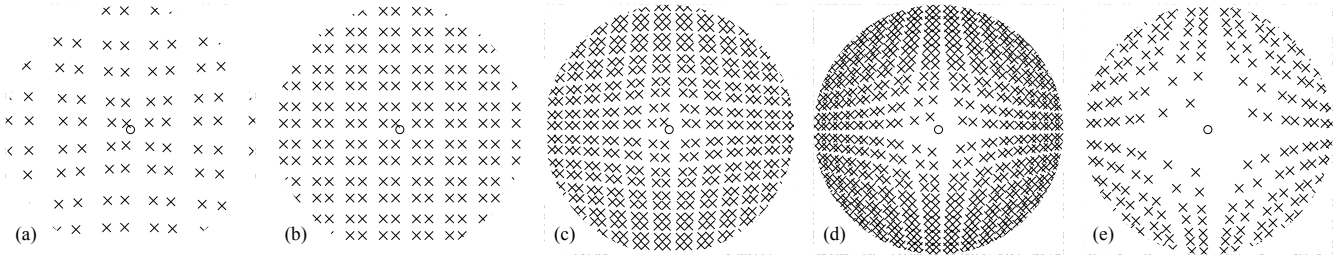


Figure 6: Diagrams of 2-D virtual source lattices radially transposed. The resultant temporal echo density profiles correspond to: a) 1.7 dimensions, b) 2.0 dimensions (original), c) 2.5 dimensions, d) 3.3 dimensions, e) 4.2 dimensions.

6. FRACTIONAL DIMENSIONS AND LATTICE TRANSPOSITION

It is possible to transpose a lattice of virtual sources from an n -D temporal echo density to an m -D temporal echo density. This can shape the pseudorandom behavior of the echoes and result in an IR with properties of an m -dimensional lattice. In this case, m can be any positive number including non-integers. The approach that follows is to generate a lattice of virtual sources and to then transpose them radially around the receiver.

Given an n -D lattice, each virtual source has a relative radial location with respect to the receiver. This radius is given by Δr_n . To obtain a transposition function, assume that $E_n = E_m$, where E is given by Eq. 11 and m is the desired dimensionality. Next, within the formulas for E_n and E_m , replace the variables $c^n t^n$ and $c^m t^m$ with the equivalent Δr_n^n and Δr_m^m , respectively. This results in

$$\frac{\pi^{\frac{n}{2}} \Delta r_n^n}{\Gamma(\frac{n}{2} + 1) V_{rn}} = \frac{\pi^{\frac{m}{2}} \Delta r_m^m}{\Gamma(\frac{m}{2} + 1) V_{rm}} \quad (15)$$

Solving Eq. 15 for Δr_m results in the transposition function

$$\Delta r_m = \eta \Delta r_n^{\frac{n}{m}} \quad (16)$$

where Δr_m is the new radius and η is a constant given by

$$\eta = \left(\frac{\pi^{\frac{n}{2}} \Gamma(\frac{m}{2} + 1) V_{rm}}{\pi^{\frac{m}{2}} \Gamma(\frac{n}{2} + 1) V_{rn}} \right)^{\frac{1}{m}} \quad (17)$$

Radially transposed image lattices, along with the original 2-D lattice, are shown in Fig. 6.

7. RESULTS

7.1. Computational Complexity

The fast image method is demonstrably faster than the Allen and Berkley algorithm [40][11]. There is limited discussion in the literature, however, on the theoretical basis for this. There has also never before been an extension to n -D.

7.1.1. Mathematical and Algorithmic Theory

Computation time in existing algorithms [23][11] is dependent on many factors. Much of the computational costs result from calculating the distances to the virtual sources. Distance calculations for sources outside the "sphere of interest" are unnecessary and can be

omitted [11]. This omission leads to a theoretical speedup factor for the distance calculations that is given as the ratio of the volume of an n -cube over the volume of an inscribed n -sphere. This is given as

$$s_{nd} = \frac{2^n \Gamma(\frac{n}{2} + 1)}{\pi^{\frac{n}{2}}} \quad (18)$$

Some values of s_{nd} are given in Table 2. When the number of dimensions is $n = 20$, the number of distance calculations is reduced by a factor of 4.1×10^7 , and thus the total computation time ought to be dramatically reduced.

Table 2: Reduction factor for the number of distance calculations in n -D space.

n	s_{nd}
1	1.0000 e +00
2	1.2732 e +00
3	1.9099 e +00
4	3.2423 e +00
5	6.0793 e +00
6	1.2385 e +01
7	2.7091 e +01
8	6.3074 e +01
9	1.5522 e +02
10	4.0154 e +02
11	1.0870 e +03
12	3.0676 e +03
13	8.9960 e +03
14	2.7340 e +04
15	8.5905 e +04
16	2.7848 e +05
17	9.2971 e +05
18	3.1912 e +06
19	1.1246 e +07
20	4.0632 e +07

The Allen and Berkley algorithm calculates the total reflection coefficient as

$$\beta_{total} = \beta_{x_1,1}^{|i_1-u_1|} \beta_{x_1,2}^{|i_1|} \beta_{x_2,1}^{|i_2-u_2|} \beta_{x_2,2}^{|i_2|} \beta_{x_3,1}^{|i_3-u_3|} \beta_{x_3,2}^{|i_3|} \dots \beta_{x_n,1}^{|i_n-u_n|} \beta_{x_n,2}^{|i_n|} \quad (19)$$

where the i 's and the u 's are indices corresponding to the room index. This requires $2n$ multiplies, $2n$ exponentiations, and n subtractions. Each exponentiation can be calculated by either a product summation or by using the identity

$$\beta_x^{|i|} = e^{|i| \ln \beta} \quad (20)$$

which is classically expressed as a series expansion.

In contrast, the algorithm in [11], calculates the total reflection coefficient as

$$\beta_{total} = \beta_{x_1, i_1} \beta_{x_2, i_2} \beta_{x_3, i_3} \dots \beta_{x_n, i_n} \quad (21)$$

where the value of each $\beta_{x,i}$ is taken from a look-up table. This requires n multiplies, 0 exponentiations, and 0 subtractions. While some computation time is needed to produce the look-up tables, it is small and only becomes significant when the calculation is limited to low order reflections.

Computation time is also required when calculating the pressure amplitude distance relation given in Eq. 8. The denominator has the term r^{n-1} . Because $n - 1$ is always an integer, the value of the exponentiation can be calculated by either a series of multiplies or as a series expansion of Eq. 20. By using the former, total computation time was reduced by around one half for calculations in 6-D.

7.1.2. Benchmarks

Both the Allen and Berkley algorithm and the fast image method were programmed for a fixed number of dimensions. This effort was continued until both algorithms had been programmed for all dimensions from one up through eight. Each of the 16 algorithms ran within a small separate standalone executable. Benchmarks were performed on a Linux system, and the `time` command was used to measure program run-time. Each program was set to loop its respective algorithm a fixed number of times so that the total run-time was at least one second. The average algorithm run-time was then calculated and used to find the speedup factor.

When considering program run-time, the time-length of the RIR filter should be considered relative to the room volume. In fact, time-length and room volume can be used to accurately estimate the total number of image points, and this will correlate very well with total execution time. Time-length is an important property of an RIR, and for this reason, benchmarks were made with time-length as an input variable. Because different time-lengths cause the number of mathematical operations to vary somewhat, benchmarks were performed for two time-lengths: t_1 and $2t_1$.

The results of the tests are listed in Table 3. The speedup factor for time-length t_1 starts at 1.5 for 1-D and steadily increases to 316.0 for 8-D. For time-length $2t_1$ the results were similar with a factor of 1.6 for 1-D and 210.0 for 8-D. The speedup factors obtained for 3-D were similar to those obtained in [11] and [40].

Both the number of image points and the computation time increased greatly with the number of dimensions. For the 8-D Allen and Berkley algorithm with time-length $2t_1$, the algorithm run-time was nearly three hours. It's expected that benchmarks in dimensions higher than eight would consume an unreasonable amount of time.

In theory, the outputs from the two algorithms are identical. In practice, there is a slight difference due to the way that each implementation produces floating-point round-off error. The difference, however, is small and the two outputs can still be validated against one another.

Table 3: Speedup factor s_n for RIRs of length t_1 and $2t_1$.

n	t_1		$2t_1$	
	s_n	image points	s_n	image points
1	1.5	7	1.6	17
2	1.8	57	2.0	236
3	6.1	311	7.1	2,725
4	23.6	1,618	13.5	28,033
5	70.2	7,929	30.8	262,155
6	133.0	35,660	64.3	2,262,971
7	230.6	147,327	147.7	18,206,694
8	316.0	562,993	210.0	137,616,917

7.2. Evaluation

Using the methods laid out in this paper, impulse responses of rooms from one to twelve spatial dimensions have been calculated. Calculations in dimensions higher than twelve are also possible.

By transposing the locations of virtual sources, n -D lattices have been forced to produce the same temporal echo densities as both higher and lower dimensional lattices. Image lattices have also been transposed to fractional dimensions, and the range of this extends below 1-D.

Calculated RIRs have been convolved with audio signals to perform n -D room simulations. All simulations sounded similar to reverberation. Simulations in 1-D produced delay type effects while simulations that had been scaled to dimensions near 1-D produced exotic delay type effects. In 2-D, they produced a reverberation effect and the RIR had plate-like properties. It is interesting to note that calculations in the frequency domain could model frequency dispersion and this would result in RIRs with even more plate-like properties. In 3-D, RIRs produced reverberant-like effects. In dimensions above three, the effects were also reverberant like. It was noted that decay times seemed to drop off more abruptly, although a rigorous numerical analysis would be more informative.

8. CONCLUSIONS

The image method has been generalized to n -D. Acoustic spaces up to 12-D have been simulated, and the behavior of fractional dimensions has been modeled. Strong similarities between delay, plate, and room reverberation have been illustrated.

Computation time increases significantly with each additional dimension. Use of the fast image method dramatically reduced the number of computations. Its use became critical for higher dimensional calculations where computation time became an issue.

9. REFERENCES

- [1] P. Huang, J.S. Abel, "Aspects of reverberation echo density," in *Audio Eng. Soc. Convention (123)*, New York, United States, Oct. 5-8, 2007, p. 7163.
- [2] J.S. Abel, P. Huang, "A simple, robust measure of reverberation echo density," in *Audio Eng. Soc. Convention (121)*, San Francisco, United States, Oct. 5-8, 2006, p. 6985.

- [3] P. Huang, J.S. Abel, H. Terasawa, J. Berger, "Reverberation echo density psychoacoustics," in *Audio Eng. Soc. Convention (125)*, San Francisco, United States, Oct. 2-5, 2008, p. 7583.
- [4] J. Parker, S. Bilbao, "Spring reverberation: A physical perspective," in *Proc. Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 1-4, 2009.
- [5] S. Bilbao, J. Parker, "A virtual model of spring reverberation," *IEEE Trans. Audio, Speech, Language Processing*, vol. 18, no. 4, 2010.
- [6] K. Arcas, A. Chaigne, "On the quality of plate reverberation," *Appl. Acoust.*, vol. 71, no. 2, pp. 147-156, 2010.
- [7] S. Bilbao, L. Savioja, J.O. Smith III, "Parameterized finite difference schemes for plates: Stability, the reduction of directional dispersion and frequency warping," *IEEE Trans. Audio, Speech, Language Processing*, vol. 15, no. 4, 2007.
- [8] F. Avanzini, *Algorithms for Sound and Music Computing*, chapter 4, Dec 2008.
- [9] C. Borß, "A VST reverberation effect plugin based on synthetic room impulse responses," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 1-4, 2009.
- [10] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low- and mid-frequency room acoustics," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10, 2010.
- [11] S.G. McGovern, "Fast image method for impulse response calculations of box-shaped rooms," *Appl. Acoust.*, vol. 70, no. 1, pp. 182-189, 2009.
- [12] A. Kulowski, "Algorithmic representation of the ray tracing technique," *Appl. Acoust.*, vol. 18, no. 6, pp. 449-469, 1985.
- [13] S. Laine, S. Siltanen, T. Lokki, L. Savioja, "Accelerated beam tracing algorithm," *Appl. Acoust.*, vol. 70, no. 1, pp. 172-181, 2009.
- [14] Z.Y. Huang, W.K. Jiang, "An effective method calculating acoustic Green's function for closed rectangular cavity using the Ewald's summation technique," *Acta Acust. Acust.*, vol. 93, no. 5, pp. 853-856, 2007.
- [15] R. Duraiswami, D.N. Zotkin, N.A. Gumerov, "Fast evaluation of the room transfer function using multipole expansion," *IEEE Trans. Audio, Speech, Language Processing*, vol. 15, no. 2, 2007.
- [16] N. Tsingos, I. Carlbom, G. Elko, R. Kubli, T. Funkhouser, "Validating acoustical simulations in the Bell Labs Box," *IEEE Comp. Graph. and Apps.*, vol. 22, no. 4, pp. 28-37, 2002.
- [17] H. Penttinen, M. Karjalainen, T. Paatero, H. Järveläinen, "New techniques to model reverberant instrument body responses," in *Proc. Intl. Computer Music Conf. (ICMC-01)*, La Habana, Cuba, Sept. 18-22, 2001.
- [18] M. Karjalainen, J.O. Smith III, "Body modeling techniques for string instrument synthesis," in *Proc. Intl. Computer Music Conf. (ICMC-96)*, Hong Kong, 1996.
- [19] C.B. Maxwell, "Real-time reverb simulation for arbitrary object shapes," in *Proc. Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 10-15, 2007.
- [20] L. Savioja, M. Karjalainen, T. Takala, "DSP formulation of a finite difference method for room acoustics simulation," in *Proc. IEEE Nordic Signal Processing Symp. (NORSIG'96)*, Espoo, Finland, Sept. 1996, pp. 455-458.
- [21] A. Kelloniemi, V. Välimäki, P. Huang, L. Savioja, "Artificial reverberation using a hyper-dimensional FDTD mesh," in *Proc. European Signal Processing Conf. (EUSIPCO-05)*, Sept. 2005.
- [22] A. Kelloniemi, V. Välimäki, P. Huang, L. Savioja, "Hyper-dimensional digital waveguide mesh for reverberation modeling," in *Proc. Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 10-15, 2007.
- [23] J.B. Allen, D.A. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Am.*, vol. 65, no. 4, pp. 943-950, 1979.
- [24] D.T. Murphy, D.M. Howard, "Modelling and directionally encoding the acoustics of a room," *Electronics Letters*, vol. 34, no. 9, pp. 864-865, 1998.
- [25] D.T. Murphy, D.M. Howard, "2-D digital waveguide mesh topologies in room acoustics modelling," in *Proc. Digital Audio Effects (DAFx-00)*, Verona, Italy, Dec. 7-9, 2000.
- [26] D.T. Murphy, A. Kelloniemi, J. Mullen, S. Shelley, "Acoustic modeling using the digital waveguide mesh," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 55-66, 2007.
- [27] J. Moorer, "About this reverberation business," *Comp. Music Journal*, vol. 3, no. 2, pp. 13-28, 1979.
- [28] M. Foco, P. Polotti, A. Sarti, S. Tubaro, "Sound spatialization based on fast beam tracing in the dual space," in *Proc. Digital Audio Effects (DAFx-03)*, London, UK, Sept. 8-11, 2003.
- [29] F. Antonacci, M. Foco, A. Sarti, S. Tubaro, "Fast tracing of acoustic beams and paths through visibility lookup," *IEEE Trans. Audio, Speech, Language Processing*, vol. 16, no. 4, 2008.
- [30] J. Brunskog, "Image solution for clamped finite beams," *J. Sound Vib.*, vol. 287, no. 4-5, pp. 1057-1064, 2005.
- [31] R. Gunda, S.M. Vijayakar, R. Singh, "Method of images for the harmonic response of beams and rectangular plates," *J. Sound Vib.*, vol. 185, no. 5, pp. 791-808, 1995.
- [32] J. Cuenca, F. Gautier, L. Simon, "The image source method for calculating the vibrations of simply supported convex polygonal plates," *J. Sound Vib.*, vol. 322, no. 4-5, pp. 1048-1069, 2009.
- [33] G. Arfken, H. Weber, *Mathematical Methods for Physicists*, p. 911, Academic Press, San Diego, CA, USA, 2001.
- [34] E.A. Lehmann, A.M. Johansson, "Prediction of energy decay in room impulse responses simulated with an image-source model," *J. Acoust. Soc. Am.*, vol. 124, no. 1, pp. 269-277, 2008.
- [35] J. António, L. Godinho, A. Tadeu, "Reverberation times obtained using a numerical model versus those given by simplified formulas and measurements," *Acta Acust. Acust.*, vol. 88, no. 2, pp. 252-261, 2002.
- [36] M. Gorzel, G. Kearney, F. Boland, H. Rice, "Virtual acoustic recording: An interactive approach," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10, 2010.

- [37] A. Kelloniemi, V. Välimäki, L. Savioja, “Simulation of room acoustics using 2-D digital waveguide meshes,” in *Intl. Conf. Acoust. Speech and Signal Processing (ICASSP-06)*, Toulouse, France, May 14-19, 2006.
- [38] K. Meesawat, D. Hammershøi, “An investigation on the transition from early reflections to a reverberation tail in a BRIR,” in *Proc. of the Int. Conf. on Auditory Display (ICAD02)*, Kyoto, Japan, July 2-5, 2002.
- [39] A. Lindau, L. Kosanke, S. Weinzierl, “Perceptual evaluation of physical predictors of the mixing time in binaural room impulse responses,” in *Audio Eng. Soc. Convention (128)*, London, United Kingdom, May 22-25, 2010, p. 8089.
- [40] E.A. Lehmann, A.M. Johansson, “Diffuse reverberation model for efficient image-source simulation of room impulse responses,” *IEEE Trans. Audio, Speech, Language Processing*, vol. 18, no. 6, 2010.
- [41] D. Halliday, R. Resnick, J. Walker, *Fundamentals of Physics*, chapter 18, John Wiley & Sons, Inc, New York, USA, fifth edition, 1997.

A. APPENDIX: N-D GEOMETRIC FORMULAS

The volume of an n -D room is given by

$$V_{rn} = \prod_{i=1}^n L_i \quad (22)$$

The volume of an n -sphere is given by

$$V_{sn} = \frac{\pi^{\frac{n}{2}} r^n}{\Gamma(\frac{n}{2} + 1)} \quad (23)$$

where r is the radius of the sphere and Γ is the gamma function. The surface area is the derivative of the volume. Using the gamma function identity

$$\frac{2}{\Gamma(\frac{n}{2})} = \frac{n}{\Gamma(\frac{n}{2} + 1)} \quad (24)$$

the surface area of an n -sphere is given by

$$S_{sn} = \frac{2\pi^{\frac{n}{2}} r^{n-1}}{\Gamma(\frac{n}{2})} \quad (25)$$

B. APPENDIX: DERIVATION OF N-D PRESSURE DISTANCE RELATION

There are a number of acoustics-related relations that can be found in general physics texts [41]. The following derivation will be based heavily around this.

The acoustic pressure amplitude is given by

$$p = v\rho\omega s \quad (26)$$

where v is the propagation speed of the wave, ρ is the mass density, ω is the angular frequency, and s is the displacement amplitude. The acoustical kinetic energy contained in an infinitesimal element of air is given by

$$dK = \frac{1}{2} dm v_s^2 \quad (27)$$

where v_s is the velocity of the oscillating air mass element dm . For a spherical wave traveling radially outward, v_s is given by

$$v_s = -\omega s \sin(kr - \omega t) \quad (28)$$

The mass of the air element is given by multiplying density ρ by the volume element. For an n -D spherical wave, the volume element is given by the product of the surface area of an n -sphere and an infinitesimal shell thickness dr . For the mass element, this results in

$$dm_n = \rho \frac{2\pi^{\frac{n}{2}} r^{n-1}}{\Gamma(\frac{n}{2})} dr \quad (29)$$

Substituting Eqs. 28 and 29 into Eq. 27 results in

$$dK = \frac{1}{2} \left(\rho \frac{2\pi^{\frac{n}{2}} r^{n-1}}{\Gamma(\frac{n}{2})} dr \right) \omega^2 s^2 \sin^2(kr - \omega t) \quad (30)$$

The rate of flow of kinetic energy can be shown to be

$$\frac{dK}{dt} = \rho \frac{\pi^{\frac{n}{2}} r^{n-1}}{\Gamma(\frac{n}{2})} v \omega^2 s^2 \sin^2(kr - \omega t) \quad (31)$$

where the propagation speed v is equal to dr/dt . It follows that the time average of the kinetic energy flow rate is given by

$$\overline{\left(\frac{dK}{dt}\right)} = \frac{1}{2} \rho \frac{\pi^{\frac{n}{2}} r^{n-1}}{\Gamma(\frac{n}{2})} v \omega^2 s^2 \quad (32)$$

where the time average of $\sin^2(kr - \omega t) = \frac{1}{2}$. The power transmitted is equal to the sum of the rates of both the kinetic energy and the potential energy. This is equivalent to two times the average kinetic energy.

$$P = \overline{\left(\frac{dK}{dt}\right)} + \overline{\left(\frac{dU}{dt}\right)} = 2 \overline{\left(\frac{dK}{dt}\right)} \quad (33)$$

Substituting Eq. 32 into 33 and simplifying results in

$$P = \rho \frac{\pi^{\frac{n}{2}} r^{n-1}}{\Gamma(\frac{n}{2})} v \omega^2 s^2 \quad (34)$$

Solving for s results in an expression for the n -D particle displacement.

$$s = \sqrt{\frac{P \Gamma(\frac{n}{2})}{\rho v \omega^2 \pi^{\frac{n}{2}} r^{n-1}}} \quad (35)$$

Substituting Eq. 35 into Eq. 26 gives

$$p = (v\rho\omega) \sqrt{\frac{P \Gamma(\frac{n}{2})}{\rho v \omega^2 \pi^{\frac{n}{2}} r^{n-1}}} = \sqrt{\frac{v\rho P \Gamma(\frac{n}{2})}{\pi^{\frac{n}{2}} r^{n-1}}} \quad (36)$$

Finally, setting

$$A = \sqrt{\frac{v\rho P \Gamma(\frac{n}{2})}{\pi^{\frac{n}{2}}}} \quad (37)$$

Eq. 36 can be rewritten as

$$p(r)_n = \frac{A}{\sqrt{r^{n-1}}} \quad (38)$$

In Eqs. 1 and 3, the constant A_q is assumed to include an additional factor for the total reflection coefficient.

COMPUTATIONALLY EFFICIENT HAMMOND ORGAN SYNTHESIS

Jussi Pekonen, Tapani Pihlajamäki, and Vesa Välimäki

Department of Signal Processing and Acoustics
Aalto University School of Electrical Engineering
Espoo, Finland

Jussi.Pekonen@aalto.fi, Tapani.Pihlajamaki@aalto.fi, Vesa.Valimaki@tkk.fi

ABSTRACT

The Hammond organ is an early electronic musical instrument, which was popular in the 1960s and 1970s. This paper proposes computationally efficient models for the Hammond organ and its rotating speaker system, the Leslie. Organ tones are generated using additive synthesis with appropriate features, such as a typical fast attack and decay envelope for the weighted sum of the harmonics and a small amplitude modulation simulating the construction inaccuracies of tone wheels. The key click is realized by adding the sixth harmonic modulated by an additional envelope to the original organ tone. For the Leslie speaker modeling we propose a new approach, which is based on time-varying spectral delay filters producing the Doppler effect. The resulting virtual organ, which is conceptually easy, has a pleasing sound and is computationally efficient to implement.

1. INTRODUCTION

The Hammond organ was one of the very first electronic synthesizers constructed in the early 20th century. Originally it was designed to be a low-cost substitute to the pipe organs used in churches, but later in the 1960s and 1970s it also became a commonly used keyboard instrument in popular music productions due to its characteristic timbre. This special timbre of the Hammond organ was partly produced by the speaker that was used for the reproduction of the organ sound. This reproduction device, the Leslie speaker, had two rotating units and its original target was to simulate the continuously shifting sound sources emanating from the different parts of the broad wall of pipes of the church organs.

Due to the difficulties in transporting the heavy organ unit (almost 200 kilograms) and the Leslie speaker and the possible technical failures of the electromechanical instrument, musicians have craved for a more easily portable unit that can be operated reliably. To meet this demand, many electronic and digital keyboards have a readily available tone that imitates the sound of the Hammond organ. On the other hand, the sound of other organs that did not necessarily imitate the Hammond organ have also been emulated, see e.g. [1]. In addition, the effect produced by the Leslie speaker is still widely used and devices that emulate the processing performed by the speaker are also available.

In this paper, computationally efficient models for both the Hammond organ and the Leslie speaker are presented. The sound

generation principle used in the organ is presented Section 2 and its discrete-time replication is discussed. Section 3 presents the proposed Leslie rotating speaker effect model that is based on time-varying spectral delay filters (SDFs). The computational complexity of the models is discussed in Section 4. Section 5 concludes the paper.

2. ORGAN MODEL

The Hammond organ was originally designed to be a more compact version of the Telharmonium synthesizer [2]. Similarly to the Telharmonium, the Hammond organ mixed the sound of several electrical tone generators, tone wheels, to synthesize a desired sound [3]. Each tone wheel consisted of a metallic disk having a grooved rim in the proximity of a magnetic pickup. When the disk was rotated, the grooves of the rim induced an alternating current to the coil of the pickup. The generated current was sinusoid-like, and the frequency of the sinusoid depended on the number of grooves and the speed of rotation [3].

For each key in the manual (in practice, a keyboard) seven tone wheels were assigned and they produced the first, the second, the third, the fourth, the fifth, the sixth, and the eighth harmonic of the frequency corresponding to the musical note associated with the key [3]. Since there were only a limited number of tone wheels built inside the organ chassis, the produced harmonics were not always exact multiples of the fundamental frequency. In such cases, the tone wheel that produced the closest frequency to the harmonic was selected [3]. In the later production models of the Hammond organ, the number of tone wheels associated with a key was increased to nine. The added components were the octave below the fundamental frequency (0.5 times the fundamental) and the musical fifth (1.5 times the fundamental).

The amplitudes of the harmonics were controlled separately with a set of sliding controllers, drawbars [3]. The original organ design contained two hand-operated manuals for both of which an individual drawbar was associated. In addition, the original design contained a pedal manual that had a separate drawbar with only two controllers [3]. In the later production models, the drawbar of the pedal manual was merged with the other drawbars to create two nine-bar controllers. In the early models, the drawbars had discrete values ranging from zero (off) to eight (fully on). Later, continuous-value controllers were introduced to provide a more flexible control. In addition to the manual mixing of the generated harmonics, the original design of the Hammond organ included a set of preset switches that set the amplitudes of the harmonics to predefined values [3].

The Hammond organ also contained a tremolo effect unit that could be applied to the resulting sound after the mixing stage [3].

This work was partly supported by the Academy of Finland, project numbers 122815 and 121252. The work of the second author has been partially funded by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° [240453].

The frequency of the tremolo effect, five Hertz, was set by the synchronous motor rotating the tone wheels. The depth of the tremolo effect could be controlled continuously, and the effect was more pronounced at high frequencies than at low frequencies due to the construction of the effect [3].

2.1. Digital modeling of the Hammond organ

According to the description given above, a simplified digital Hammond organ model can be constructed using the additive synthesis technique. The number of sinusoids is directly set by the number of tone wheels dedicated for a key. However, it should be noted that since the amplitude of the sinusoids is controlled by the drawbar, some of the sinusoids can have an amplitude that is so weak that they are not reasonable to be synthesized at all. In addition to the sinusoids required for the actual organ tone synthesis, a sinusoid with frequency of five Hertz can be included to the model to simulate the tremolo effect.

Since the drawbars were used to control the timbre of the produced sound by setting the individual amplitudes, there are numerous possible combinations even when the discrete-valued drawbars are used. Of these possible combinations, a mixture that consists of four components is considered as an example timbre in this paper. The (exact) harmonic components used in the example tone are the first, the second, the third, and the eighth when the fundamental frequency is above 130 Hz (note C3). Below that frequency the first and the eighth harmonic are used together with the sub-octave harmonic and the musical fifth. The notes above the given frequency can be understood to be played with one (hand-operated) manual whereas the notes below that frequency are played with another manual and/or with the pedal manual. These components were manually chosen to produce a pleasant tone with the manually chosen mixing setup given below.

The component amplitudes of the example timbre are 1 (full amplitude), 0.2, 0.2, and 0.1, ordered from the lowest frequency to the highest in both cases. As the general amplitude envelope an envelope that has a rapid (order of a few milliseconds) attack and release is used. After the attack phase of the amplitude envelope, the envelope can have a fast decay (a constant decay time of 50 ms) to a sustain level that is quite close to the maximum amplitude. With this decay simulation, the initial burst of current at the synthesizer output introduced by the rapid pressing of a key [3] can be simulated. An example of the general amplitude envelope is plotted in Figure 1(a).

2.2. Model extensions

The model described above can be extended in a couple of ways. First, the noise-like signal produced by the closing and the opening of the tone generation circuits at the events of key pressing and release, respectively, can be added to the generated sound at the respective events. The noise produced by these events has most of its energy at high frequencies, and especially the "key click" produced by the key pressing is quite pronounced and it is often used by musicians due to its percussive effect it adds to the tone. However, instead of simulating the key click by filtering a noise signal, it can be approximated by adding the sixth harmonic component with an additional amplitude envelope to the output signal. The amplitude envelope of the click signal has again a rapid attack and release (the same values as with the main tone can be used) and a fast decay (a constant decay time of 70 ms) to the zero level,

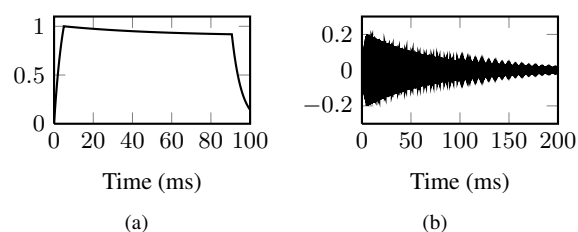


Figure 1: (a) Amplitude envelope of the example timbre. (b) Time-domain plot of the proposed "key click" effect.

as indicated in the time-domain plot shown in Figure 1(b). This manually chosen approximation effectively adds a click-like component to the tone and it is computationally efficient to implement.

Second, since the tone wheels were mechanical devices, the rotating disks cannot be constructed to be perfectly symmetric. These imperfections in the symmetry can be included to the model with a small sinusoidal amplitude modulation (at the frequency of the tone wheel rotation and the tremolo effect, five Hertz) that simulates the small distance fluctuations between the disk and the pickup. Similarly, the imperfections in the grooving of the disk rim will produce the harmonics of the nominal frequency of the wheel.

3. ROTATING SPEAKER EFFECT

The Leslie rotating speaker is constructed from two separate rotating units, the treble unit and the bass unit [4]. The treble unit is formed by a symmetrical dual-horn structure which is rotated with a motor. Only one of these horns is used for the sound reproduction with the other acting as a dummy for symmetrical mass and form. These horns are quite directive and thus the output effect contains significant amplitude modulation. Furthermore, the location of the sound source is at the mouth of the rotating horn. This means that the distance of the horn from the listener is changing resulting in a frequency modulation effect.

The bass unit is constructed with a stationary loudspeaker with produced sound fed through a rotating wooden drum. This drum effectively adds a directional pattern to the bass sound and thus generates amplitude modulation when rotated. Henriksen [4] also mentions that it is possible that there is a frequency modulation effect present near the crossover frequency of the two units although the amplitude modulation is the dominant effect.

To create a similar effect compared to the rotating speaker both frequency modulation and amplitude modulation should be applied similarly to the real speaker. Amplitude modulation is simple to implement if the frequency-dependent directivity of the loudspeaker is dismissed. Then it is enough to multiply the signal with a sinusoidal oscillator scaled appropriately for the desired depth of effect.

The frequency modulation of an arbitrary signal can be produced by feeding the signal into a delay line and by modulating the delay-line length [5]. However, in order to produce a smooth effect, fractional-delay filters (like the linear interpolator) [6] have to be applied, which makes the implementation more complex. An alternative efficient solution is to apply spectral delay filters (SDF) for this purpose.

Several studies have previously been published about the mod-

eling of the Leslie rotating speaker. Smith et al. [7] simulated the Doppler effect with interpolated delay lines and used Leslie simulation as an example case. Their implementation is comparable to the one presented here. Kronland-Martinet and Voinier [8] provided a study on the perceptual simulation of moving sources and, similarly to Smith et al., used the Leslie speaker as one of their application examples. Herrera et al. [9] approached the task by measuring impulse responses in a dense rotational pattern for both units. They synthesized the effect by applying a time-varying FIR filter.

3.1. Spectral delay filters

Spectral delay filters [10] are a recently proposed method for producing a frequency-dependent delay with a first-order allpass filter chain. One of their advantages is that the distribution of the delay in frequency can be controlled with the filter coefficient. The spectral delay filter is formulated by starting from a first-order allpass filter with transfer function

$$H(z) = \frac{a_1 + z^{-1}}{1 + a_1 z^{-1}}, \quad (1)$$

where $a_1 \in [-1, 1]$ is the filter coefficient.

The magnitude response of this filter is 0 dB at all frequencies. The phase response is nonlinear (unless $a_1 = 0$) and thus the group delay depends on frequency. The group delay of a single allpass section can be calculated with

$$\tau_g(\omega) = \frac{1 - a_1^2}{1 + 2a_1 \cos \omega + a_1^2}. \quad (2)$$

With absolute values of filter coefficient a_1 close to unity, the delay can be quite large at some frequencies. For example, with $a_1 = -0.9$, the delay at low frequencies is close to 19 samples.

As SDF is a cascade of first-order allpass filters, the transfer function an SDF is

$$H(z) = \left(\frac{a_1 + z^{-1}}{1 + a_1 z^{-1}} \right)^N, \quad (3)$$

where N is the length of the filter cascade. This effectively multiplies the amount of produced delay while keeping the control of delay distribution simple.

Pekonen *et al.* [11] and Kleimola *et al.* [12] studied the possibility of varying the filter coefficient of an SDF through time. It was noted that the stability conditions for time-varying SDFs are the same as with the single allpass filter, i.e. $|a_1| \leq 1$, if the condition holds for every time step n [11]. Here, the difference equation of the Direct form I realization,

$$y(n) = m(n)x(n) + x(n-1) - m(n)y(n-1), \quad (4)$$

where $x(n)$ is the input signal, $y(n)$ is the output signal, and $m(n)$ is the modulator signal, is used [13, 12].

The modulator signal, and how it is applied, can significantly affect the produced effect. For example, a sinusoidal modulator produces different effects if it varies between -1 and 1 or between -1 and -0.9 . Thus, for this paper, two additional variables are defined for controlling the modulation. This is done by performing the substitution

$$m(n) = M_s m_0(n) + M_b \quad (5)$$

to (4). Here, $m_0(n) \in [-1, 1]$ is the unscaled modulator signal, M_s is a modulator scaling term, and M_b is a modulator bias term.

3.2. Implementation

Based on the construction of the Leslie speaker, a model for implementation can be created. A block diagram of this model is shown in Figure 2.

The modulating signal should be sinusoidal as that represents the movement of the real speaker best. As there are separate motors for the treble unit and the bass unit in the real speaker, it is prudent to also use two separate oscillators for modulation. This enables the use of the characteristic speed-up and slowdown effects of the Leslie speaker where the bass unit accelerates slower due to the inertia of the unit. This acceleration mismatch is simple to implement with different modulator frequency envelopes.

The highpass and lowpass filters can be created with any suitable filter implementation although a digital version of the real crossover filters would be preferable. In the demonstration implementation, these filters were fourth-order digital Butterworth IIR filters with the cutoff frequency at 800 Hz.

Amplitude modulation can be implemented directly by multiplication with a scaled modulator. This scaling is done so that the values of the sinusoid are between one and α , where α is a value between 0 and 1.0. In the demonstration implementation, α was selected to be 0.9 for both paths.

The frequency modulation is performed with two SDFs. The parameters of the SDFs differ between the treble and the bass pathways to optimize the effect on both pathways. The parameters presented in Table 1 were found to produce a pleasant effect.

The modulator frequency f_m for slow rotation speed was 2 Hz, and for the fast speed was 6 Hz. The modulator of the treble pathway had 0.1 Hz higher frequency compared to the bass pathway. This difference is introduced to model the motors running at similar but slightly different speeds, as described by Henriksen [4] mentioned that the speeds of the real motors are almost but not exactly the same. It was found that the effect is slightly more interesting when there is this mistuning present in the modulator frequencies.

4. DISCUSSION

The organ model presented in Section 2 provides an imitation of the original Hammond organ design. In general, the computational load of the model is rather low as it requires only ten sinusoidal oscillators when each of the nine tone components are generated separately. The tenth oscillator is dedicated for the tremolo effect. If the key click is included, one can use the output of the oscillator that produces the sixth harmonic. For the example timbre described in Section 2, only seven oscillators are required in total. It should be noted, however, that the actual computational load, i.e. the number of cycles used by the processor, depends on how the sinusoidal oscillators are implemented and that the different sine oscillator implementations may be computationally more costly in some processors than in other processors.

The number of oscillators can obviously be decreased by tabulating a predefined mixture of sinusoids into a wavetable. However, since there are numerous possible mixtures, the memory consumption of the wavetable-based approach would be huge. Therefore, for the general Hammond organ model the tones are generated more efficiently by synthesizing each of the sinusoids separately. Yet, for a small set of predefined mixtures that are not modified afterwards, like in the case of the example timbre of this

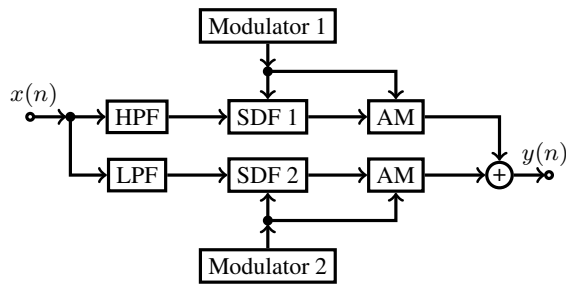


Figure 2: Block diagram of the rotator effect.

Table 1: Parameter values for the SDFs in Figure 2.

Parameter	Treble (SDF1)	Bass (SDF2)
SDF length N	4	3
Modulator scaler M_s	0.2	0.04
Modulator bias M_b	-0.75	-0.92

paper, the wavetable-based approach does provide a computationally efficient implementation.

The Leslie model presented in Section 3 produces a perceptually pleasant rotator effect. However, the model is not complete and could be extended in various ways, e.g. modeling the cabinet and the loudspeaker elements, and low-complexity methods should be implemented for them in the future. Nonetheless, the advantages of this implementation are evident. The computational complexity is quite low as the filters are of low order and otherwise there are only a few required operations. Furthermore, the implementation is simple as the same oscillators can directly modulate the amplitude of the signal and the filter coefficients of the SDFs. Only proper scaling is required to produce the desired effect.

5. CONCLUSION

A computationally efficient model of the Hammond organ and the Leslie rotating speaker were presented. The original design of the Hammond organ was described and its digital implementation was discussed. A practical example setup for the organ timbre was given, and extensions to the model were discussed. A perception-based model of the Leslie rotating speaker was proposed based on time-varying spectral delay filters (SDFs). Practical parameters for the SDFs were presented and modeling of the two operating speeds of the original speaker was discussed. In addition, the computational load of both the Hammond organ model and the Leslie rotating speaker was discussed and analyzed.

As stated above, the proposed digital Hammond organ model is a simplification of the original electromechanical device. The model neglects the crosstalk, or tone leakage, of the tone wheels to their neighboring pickups. This feature can be included by adding the sinusoids generated by the neighboring tone wheels with relatively small amplitudes to the tone wheel output signal (that is, the sinusoid). However, it should be noted that the frequencies of the crosstalk components depend on the ordering of the tone wheels, and therefore no general rules can be given for the tone leakage modeling. Furthermore, this feature can increase the computa-

tional complexity of the algorithm quite much and was therefore not included in the proposed model in the first place. In addition, the proposed organ model does not include the imperfections in the disk rim grooving.

Sound examples of the proposed organ and speaker models can be found online at <http://www.acoustics.hut.fi/go/dafx11-hammond/>.

6. REFERENCES

- [1] S. Savolainen, "Emulating a combo organ using Faust," in *Proc. 8th Linux Audio Conf.*, Utrecht, The Netherlands, May 2010.
- [2] C. Roads, *Computer Music Tutorial*, pp. 134–136, The MIT Press, Cambridge, MA, 1995.
- [3] L. Hammond, "Electronic musical instrument," U.S. Patent 1,956,350, Jan. 1934.
- [4] C. A. Henricksen, "Unearthing the mysteries of the Leslie cabinet," *Recording Engineer/Producer Mag.*, Apr. 1981, Available online <http://www.theatreorgans.com/hammond/faq/mystery/mystery.html> (date last viewed July 1, 2011).
- [5] V. Lazzarini, J. Timoney, and T. Lysaght, "The generation of natural-synthetic spectra by means of adaptive frequency modulation," *Computer Music J.*, vol. 32, no. 2, pp. 9–22, Summer 2008.
- [6] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay – tools of fractional delay filter design," *IEEE Signal Process. Mag.*, vol. 13, no. 1, pp. 30–60, Jan. 1996.
- [7] J. O. Smith, S. Serafin, J. S. Abel, and D. Berners, "Doppler simulation and the Leslie," in *Proc. COST-G6 Conf. Digital Audio Effects (DAFx-02)*, Hamburg, Germany, Sept. 2002, pp. 13–20.
- [8] R. Kronland-Martinet and T. Voinier, "Real-time perceptual simulation of moving sources: application to the Leslie cabinet and 3D sound immersion," *EURASIP J. Audio, Speech, and Music Process.*, vol. 2008, Article ID 849696, 10 pages, 2008.
- [9] J. Herrera, C. Hanson, and J. S. Abel, "Discrete time emulation of the Leslie speaker," in *Proc. 127th AES Convention*, New York, Oct. 2009, preprint no. 7925.
- [10] V. Välimäki, J. S. Abel, and J. O. Smith, "Spectral delay filters," *J. Audio Engineering Society*, vol. 57, no. 7/8, pp. 521–531, July/Aug. 2009.
- [11] J. Pekonen, V. Välimäki, J. S. Abel, and J. O. Smith, "Spectral delay filters with feedback and time-varying coefficients," in *Proc. 12th Intl. Conf. Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 2009, pp. 157–164.
- [12] J. Kleimola, J. Pekonen, H. Penttinen, V. Välimäki, and J. S. Abel, "Sound synthesis using an allpass filter chain with audio-rate coefficient modulation," in *Proc. 12th Intl. Conf. Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 2009, pp. 305–312.
- [13] V. Lazzarini, J. Timoney, J. Pekonen, and V. Välimäki, "Adaptive phase distortion synthesis," in *Proc. 12th Intl. Conf. Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 2009, pp. 28–35.

STRUCTURED SPARSITY FOR AUDIO SIGNALS

Kai Siedenburg^{*†}, Monika Dörfler[†]

^{*} Department of Mathematics, Humboldt University Berlin

[†]NuHAG, Faculty of Mathematics, University of Vienna, Austria

kai.siedenburg@gmx.de, monika.doerfler@univie.ac.at

ABSTRACT

Regression problems with mixed-norm priors on time-frequency coefficients lead to structured, sparse representations of audio signals. In this contribution, a systematic formulation of thresholding operators that allow for weighting in the time-frequency domain is presented. The related iterative algorithms are then evaluated on synthetic and real-life audio signals in the context of denoising and multi-layer decomposition. Further, initial results on the influence of the shape of the weighting masks are presented.

1. INTRODUCTION

Most audio signals of importance for humans, in particular speech and music, are highly structured in time and frequency. Typically, salient signal components are sparse in time (or frequency) and persistent in frequency (or time). Sparsity in time is connected to transient events, while sparsity in frequency is observed in harmonic components. Processing sound signals with time-frequency dictionaries is ubiquitous. The sparse structure usually seen can be further enhanced by procedures such as basis pursuit [1] or ℓ^1 -regression [2]. In the context of time-frequency dictionaries, a natural step beyond classical sparsity approaches is the introduction of sparsity criteria which take into account the two-dimensionality of the time-frequency representations used. Mixed norms on the coefficient arrays make it possible to enforce sparsity in one domain and diversity and persistence in the other domain. Regression with mixed-norm priors was first proposed in [3]. In the current contribution, we consider a family of specific regression problems with ℓ^1 and ℓ^2 priors on the coefficients; the algorithms derived thereof are refined by using local neighborhood-weighting. The performance of the resulting different operators is systematically evaluated for classical signal processing tasks like de-noising and sparse multi-layer decomposition. Applications lead to quite satisfactory results in terms of measured (SNR) and listening. The presented results reflect a first step in the exploitation of structured shrinkage in the sense of *informed analysis*, i.e., using some available prior knowledge about the signal under consideration. The main contribution is the generalization and application of structured shrinkage operators [3] to representations of audio signals by frames.

2. TECHNICAL TOOLS

We seek to expand a signal $s \in \mathbb{C}^L$ in the form

$$s(n) = \sum_{k,j} c_{k,j} \varphi_{k,j}(n) + r(n), \quad n = 1, \dots, L \quad (1)$$

This work was supported by the Austrian Science Fund (FWF) project LOCATIF(T384-N13) and the WWTF project Audio-Miner (MA09-024)

where the $\varphi_{k,j}$ denote the atoms of a time-frequency dictionary Φ , $c_{k,j}$ are the expansion coefficients and r is some residual. In order to guarantee perfect and stable reconstruction of a signal from its associated analysis coefficients $c_{k,j} = \langle s, \varphi_{k,j} \rangle$, we assume that the dictionary Φ forms a frame [4]. We consider Gabor frames, which are exhaustively used in music processing, be it under a different name: in their simplest instantiation they correspond to a sampled sliding window or short-time Fourier transform. Gabor frames consist of a set of atoms $\varphi_{k,j} = M_{bj} T_{ka} \varphi$, where T_x and M_ω denote the time- and frequency-shift-operator, resp., defined by $T_x \varphi(n) = \varphi(n - x)$, $M_\omega \varphi(n) = \varphi(n) e^{\frac{2\pi i n \omega}{L}}$, and φ is a standard window function. a and b are the time- and frequency sampling constants, and $j = 0, \dots, J - 1, k = 0, \dots, K - 1$, with $Ka = Jb = L$.

We will even assume more, namely tightness of the frames in use, which means that, up to a constant which may be set to 1, we have $s = \sum_{k,j} \langle s, \varphi_{k,j} \rangle \varphi_{k,j}$, i.e., synthesis is done with the analysis window. Tight frames are easily calculated, see [5]. In the finite discrete case, the frame's atoms constitute the columns of a matrix Φ which is of dimension $L \times p$; for tight frames, we have $\Phi \cdot \Phi^* \cdot s = s$. Since we are especially interested in the redundant case $L < p$, the additional degrees of freedom are used to promote sparsity of the coefficients.

2.1. Regression with mixed norms

Sparsity of coefficients may be enforced by ℓ^1 -regression, also known as the *Lasso* [2]. Given a noisy observation $y = s + e$ in \mathbb{C}^L it finds

$$\hat{c} = \arg \min_{c \in \mathbb{C}^p} \frac{1}{2} \|y - \Phi c\|_2^2 + \lambda \Psi(c) \quad (2)$$

with penalty term $\Psi(\cdot) = \|\cdot\|_1$ and $\lambda > 0$. Since the sequence $c_{k,j}$ is ordered along two dimensions for Gabor frames, the ℓ^1 -prior Ψ in (2) may be replaced by a two-dimensional mixed norm $\ell^{p,q}$ which acts differently on groups (indexed by g in the sequel, may be either time or frequency) and their members (indexed by m):

$$\Psi(c) = \|c\|_{p,q} = \left(\sum_g \left(\sum_m |c_{g,m}|^p \right)^{q/p} \right)^{1/q} \quad (3)$$

Subsequently, the notation (g, m) will be used in reference to the group-member structure, whereas (k, j) refers to the time-frequency indices of the Gabor-expansion. In terms of $\ell^{p,q}$, we consider the cases $p = 2, q = 1$ and $p = 1, q = 2$. The former is known as *Group-Lasso (GL)* [6] (promoting sparsity in groups and diversity in members) and the latter was termed *Elitist-Lasso (EL)* in [3]: the

$\ell^{1,2}$ constraint promotes sparsity in members, only the “strongest” members (relative to an average) of each group are retained. Landweber iterations, which solve (2) in the ℓ^1 -case, [4], also yield a solution to the generalized minimization problem induced by (3), if standard soft thresholding is replaced by a generalized thresholding operator $\mathbb{S}_{\lambda,\xi}(z_{g,m}) = z_{g,m}(1 - \xi(z))^+$. Here, $\xi = \xi_{(g,m),\lambda}$ is a non-negative function dependent on the index (g,m) and λ . The solution to (2) is then given by the iterative Landweber algorithm: choosing arbitrary c^0 , set

$$c^{n+1} = \mathbb{S}_{\lambda,\xi}(c^n - \Phi^*(y - \Phi c^n)). \quad (4)$$

It was shown in [7], that the use of the thresholding operators $\mathbb{S}_{\lambda,\xi}$, defined via ξ , leads to convergence of the iterative sequence (4) to the minimizer of (2):

$$p = 1, q = 1 : \xi^L(c_{g,m}) = \frac{\lambda}{|c_{g,m}|} \quad (\text{Lasso}) \quad (5)$$

$$p = 2, q = 1 : \xi^{GL}(c_{g,m}) = \frac{\lambda}{(\sum_m |c_{g,m}|^2)^{\frac{1}{2}}} \quad (\text{GL}) \quad (6)$$

$$p = 1, q = 2 : \xi^{EL}(c_{g,m}) = \frac{\lambda}{1 + M_g \lambda} \frac{\|c_g\|_1}{|c_{g,m}|} \quad (\text{EL}) \quad (7)$$

where $c_g = (c'_{g,1}, \dots, c'_{g,M_g})$ and $\{c'_{g,m'}\}_{m'}$ denotes for each group g the sequence of scalars $|c_{g,m}|$ in descendant order. M_g denotes some natural number depending on the magnitudes of coefficients in the group $(c_{g,1}, \dots, c_{g,M})^1$.

2.2. Refining the algorithms

To exploit structures in audio signals, like persistence in time or frequency, we refine the shrinkage operators introduced above for application in audio analysis. The coefficient $c_{g,m}$ (or groups of them) undergo shrinkage according to the energy of a *time-frequency neighborhood*. In contrast to the *groups* of GL and EL, the neighborhoods can be modeled flexibly, e.g., using weighting and overlap. Hence, we compose ξ with some neighborhood weighting functional η_N :

To an index $\gamma = (g,m)$ in a structured index set \mathcal{I} , we associate a (weighted) neighborhood $N(\gamma) = \{\gamma' \in \mathcal{I} : w_\gamma(\gamma') \neq 0\}$ with weights w_γ defined on \mathcal{I} such that $w_\gamma(\gamma) > 0$, $w_\gamma(\gamma') \geq 0$ for all $\gamma' \in \mathcal{I}$ and $\sum_{\gamma' \in N(\gamma)} w_\gamma(\gamma')^2 = 1$. Then, with $\eta_N(c_\gamma) = (\sum_{\gamma' \in N(\gamma)} w_\gamma(\gamma')^2 |c_{\gamma'}|^2)^{1/2}$, we obtain the generalized shrinkage operators by setting²

$$\begin{aligned} \xi^{WGL} &= \xi^L \circ \eta_N \quad (\text{windowed GL (WGL)}), \\ \xi^{PEL} &= \xi^{EL} \circ \eta_N \quad (\text{persistent EL (PEL)}), \\ \xi^{PGL} &= \xi^{GL} \circ \eta_N \quad (\text{persistent GL (PGL)}) \end{aligned}$$

in (5)-(7). These generalized shrinkage operators are not associated to a simple convex penalty functional, cp. [3]. Convergence properties of their Landweber-iterations are currently under study, and numerical experiments suggest convergence.

¹Cp. [7] for a more involved, but exact definition of the M_g in EL.

²[3] introduces WGL as generalization of GL while from a formal point of view it would be more appropriate to call it windowed Lasso. Nonetheless, we stick to the former nomenclature.

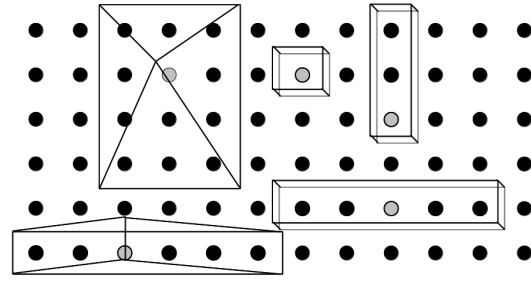


Figure 1: Sketch of different shapes of the parameterization of the neighborhoods in a schematic time-frequency plane. Rectangular and triangular (“tent”-like) windows were implemented.

3. SIMULATIONS

The generalized shrinkage operators were implemented in MATLAB with the following parameterization of the neighborhoods: For each time-frequency-index (k,j) and a neighborhood size vector $\sigma = (\sigma_1, \dots, \sigma_4)$, the neighborhood N_σ is defined as the set of indices $N_\sigma(k,j) = \{(k',j') : k' \in \{k - \sigma_4, k + \sigma_2\}, j' \in \{j - \sigma_3, j + \sigma_1\}\}$. Neighborhoods of indices close to a border of the time-frequency plane are obtained by mirroring the index set at the respective border. Rectangular and triangular weighting of the neighborhoods was implemented, with rectangular weighting only in section 3.1 and 3.2. In the plots, an index after the operator’s abbreviation specifies the group-label as time or frequency (not needed for Lasso and WGL), e.g., PEL-t signifies that the group in the respective elitist lasso is time. For the neighborhood-smoothed operators WGL, PEL, and PGL the neighborhood-size vector σ is given. To test the obtained variety of shrinkage operators, we used a simulated “toy”-signal consisting of a stationary, a transient and a noise part.³ The stationary part consists of four harmonics with fundamental frequency 440Hz and decreasing amplitudes. The obtained harmonics were shaped by a linear envelope in attack and decay. The transient part was simulated by 4 equidistant impulses with similarly decaying amplitudes. Finally, Gaussian white noise with SNR about 15 and 3dB was added.

Landweber iterations are known to converge very slowly and various methods of acceleration have been proposed [8]. As it was out of the scope of this paper to elaborate on these ideas, we used the basic iteration scheme (4). The iterations presented in the following were stopped after 100 steps. Then almost all of the final relative iteration errors were below 0.3%.

3.1. Structured denoising

As a first experiment the standard de-noising problem with additive Gaussian white noise was considered. We use a tight Gabor-frame with Hann window of length 1024 and hop size 256 (at sampling rate 44100Hz). We measure the operator’s performance in SNR: with the estimation’s approximate Landweber-limit c^* of (4) and $\hat{s} = \Phi c^*$, the SNR is $\text{SNR}(\hat{s}, s) = 20 \log_{10}(\frac{s}{\hat{s}-s})$. For comparison, the SNR is then plotted against the number of positive coefficients. Of the variety of possible operators, Fig-

³Corresponding sound files and more detailed visualizations are presented at the conference and on the webpage <http://homepage.univie.ac.at/monika.doerfler/StrucAudio>.

ure 2 presents the SNR curves of the best basic operators and their neighborhood smoothed counterparts (of which again the best of each type were chosen for the figure) at two different noise levels. It is obvious that for the lower noise level Lasso and WGL (with neighborhoods in frequency) perform best, where the WGL still outperforms the Lasso. Also, WGL with neighborhoods in time outperforms Lasso for the higher noise level. Yielding high SNR over a broad range of sparsity values, WGL thus seems to be a good choice for the de-noising task (and we made the same observations for “real life”-audio signals). Compared to WGL, the other operators GL, PGL, EL and PEL perform quite badly for the lower noise level. While PGL is constantly worse than GL, PEL seems to have some advantages over EL for higher sparsity levels. However, GL is surprisingly the second best operator for de-noising the toy example at the high noise level. Experiments with longer and more complex audio excerpts do not replicate this result, which is not surprising, since the structure of GL naturally promotes simple signal structures (which can be advantageous in some cases, see 3.2). Conclusively, the neighborhood smoothing seems to pay off in the de-noising task with Gaussian white noise, where the persistent operators WGL and PEL outperform their respective counterparts Lasso and EL. An exception is made by PGL, which performs in our experiments constantly worse than GL. Concerning the perceptual quality of the de-noised audio-material, the neighborhood smoothing of the modified operators promotes continuity in the coefficients and thus reduces the probability of isolated high energy coefficients and we observed less musical noise and higher perceptual audio quality, especially under WGL.

3.2. Multilayer decomposition

We continue processing the toy-example by aiming to extract the signal’s tonal and transient parts at the lower noise level (15dB SNR). For estimating the tonal layer, we use a tight Gabor frame with Hann window, window-length 4096 and hop size 1024. The transients are estimated starting from the transient layer + noise (which corresponds to the unrealistic but complexity reducing assumption of perfect tonal estimation) using short windows (256 samples, hop size 64). Table 1 and 2 present the performance of a sample of operators, again of each type a “basic” and neighborhood-smoothed one. The tables show the maximum SNR value of the estimation measured w.r.t. the “true” respective layer and the corresponding percentage of retained coefficients.

Concerning the tonal estimation displayed in Table 1, WGL with neighborhoods expanding in time perform best for the extraction of the tonal part in terms of SNR, while retaining relatively few coefficients. GL, PGL and PEL exhibit comparable SNR, but only with far more coefficients. As in the situation of de-noising above, the neighborhood-smoothing is useful for the Lasso and the Elitist-Lasso, but not for Group-Lasso.

As Table 2 shows clearly, GL with time as group-index performs best in terms of SNR for the estimation of the transient layer. This result is not very surprising since the example’s transient layer has a simple structure which supports the performance of GL. However, GL-t should be a good choice for transient extraction in more complex signals, since it yields broadband transients without extracting many horizontal (tonal) signal parts.

The next experiment addressed the decomposition of musical audio without the presence of quasi-ground truth. For this “real-life” application, the choice of sparsity level λ is always a difficult task. We chose the SNR-maximizing candidates from the simulations.

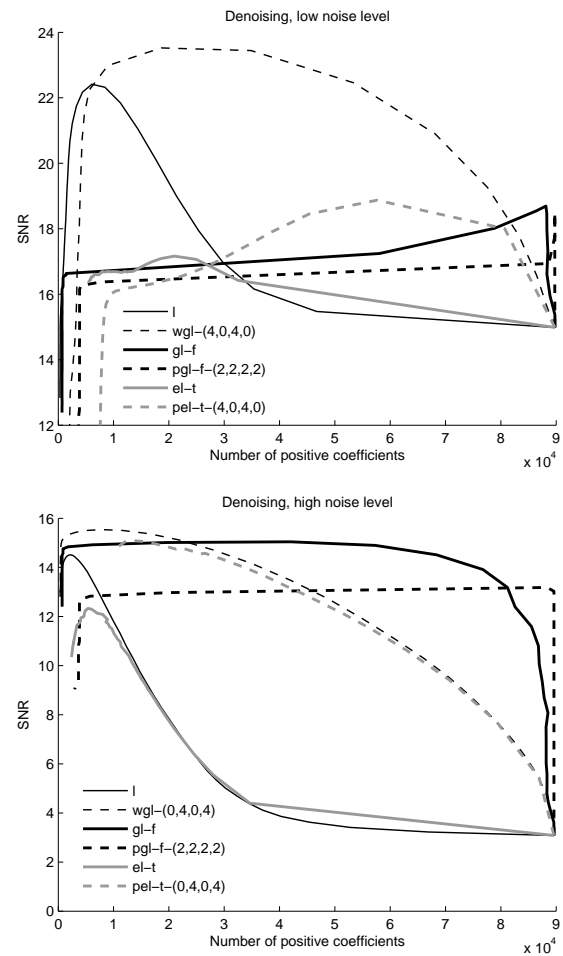


Figure 2: Overview of de-noising behavior of (modified) shrinkage operators (*l* is for Lasso). Performance measured in SNR against the number of positive coefficients at two different noise levels (15dB and 3dB).

This yielded WGL (with rectangularly weighted neighborhoods extending 4 elements in each direction of time) as estimator of the tonal layer with sparsity level $\lambda = 0.080$, and GL (with the time-index as group label) for the transient layer with $\lambda = 0.072$. We used a 5 seconds excerpt of a Jazz-record containing piano, double-bass and drums. In the decomposition, the drums (and some percussive elements of the bass) are well separated from the harmonics of piano and bass. Using GL as transient estimator works well in this example, it captures all of the soft 16th notes drum-patterns. We observed a trade-off in the choice of the sparsity level: increasing sparsity in the tonal estimation improves the separation of both layers but leads to increased damping of higher, low-energy partials of the tonal part.

3.3. Shapes

As described at the beginning of this section, the neighborhoods’ shapes (constituted by size and weighting) were implemented and parametrized in a straight-forward fashion, so far allowing for rectangular domains with either uniform (i.e. rectangular) or triangular

Table 1: Comparison of the performance of different operators in tonal estimation: maximum SNR values of estimation and “true” layer and corresponding number of retained coefficients in percent. * refers to neighborhoods (4, 0, 4, 0) while + to (0, 4, 0, 4).

Operator	Lasso	WGL ⁺	GL-f	PGL-f*	EL-t	PEL-t ⁺
max. SNR	28.7	31.2	30.5	30.7	26.2	30.6
%Coeffs	0.4	1.1	3.3	17.0	2.8	4.1

Table 2: Transient estimation: maximum SNR values of estimation and “true” layer and corresponding number of retained coefficients in percent. As above: * means (4, 0, 4, 0) and + means (0, 4, 0, 4).

Operator	Lasso	WGL*	GL-t	PGL-t ⁺	EL-f	PEL-f*
max. SNR	10.4	13.2	14.4	9.5	10.4	13.3
%Coeffs	1.0	2.9	2.2	38.9	1.4	3.7

(i.e. “tent”-like) weightings. These shapes do not necessarily have to be symmetric at the origin, as the energy of most audio signals is not symmetrically distributed around its peaks either. This fact can be exploited to feature different parts of a signal under observation. Consider Figure 3, where the iterated WGL-shrinkage results with four different neighborhood-shapes, each solely extending in time, are compared (based on a Gabor-frame with window length 1024 and overlap of 4). It is obvious that the shapes yield different (sparse) perspectives on the signal content. Whereas the symmetric neighborhoods naturally captures parts before and after the attacks (or rather time-points of maximum energy), the asymmetric ones rather retain components before (resp. after) the attacks. The orientation of the neighborhood therefore systematically promotes the preservation of different temporal segments of the signal.

4. SUMMARY AND PERSPECTIVES

We presented first results on structured sparsity approaches for Gabor frames to audio signals. Future work will focus on the convergence of the algorithms, both in a theoretical and computational setting. By taking into account methods as [9] the proposed algorithms should be accelerated significantly. On the contrary, evaluations of the algorithms’ perceptual qualities will be considered. Further, using various shapes for the weight, we aim at the extraction of more specific structures, in the sense of *sound objects* [10].

5. ACKNOWLEDGMENTS

We thank Matthieu Kowalski and the anonymous reviewers for their valuable advice.

6. REFERENCES

[1] Scott Chen, David Donoho, and Michael Saunders, “Atomic decomposition by basis pursuit,” *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

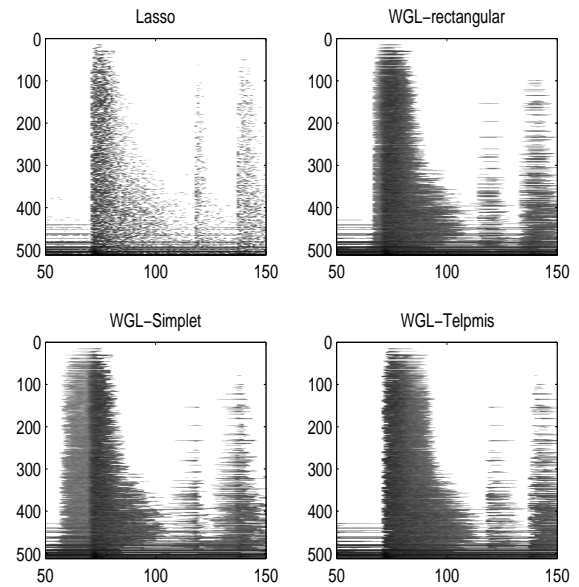


Figure 3: Iterated WGL shrinkage results for different shapes (i.e. weightings) of the neighborhood on a snare drum hit excerpt. From left to right: Lasso, Rectangular, Simplet (= simple tent, starting at 1 and then linearly decaying to zero), Telpmis (= time-reversed simple tent).

- [2] Robert Tibshirani, “Regression shrinkage and selection via the lasso,” *J. Roy. Statist. Soc. Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.
- [3] M. Kowalski and B. Torr sani, “Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients,” *Signal, Image and Video Processing*, doi:10.1007/s11760-008-0076-1, 2009.
- [4] St phane Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, 2009.
- [5] M. D rfler, “Time-frequency Analysis for Music Signals. A Mathematical Approach,” *Journal of New Music Research*, vol. 30, no. 1, pp. 3–12, 2001.
- [6] Ming Yuan and Yi Lin, “Model selection and estimation in regression with grouped variables,” *J. R. Stat. Soc. Ser. B Stat. Methodol.*, vol. 68, no. 1, pp. 49–67, 2006.
- [7] Matthieu Kowalski, “Sparse regression using mixed norms,” *Appl. Comput. Harmon. Anal.*, vol. 27, no. 3, pp. 303–324, 2009.
- [8] Ignace Loris, “On the performance of algorithms for the minimization of l1-penalized functionals,” *Inverse Problems*, vol. 25, 2009.
- [9] Amir Beck and Marc Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
- [10] G. Cornuz, L. Daudet, P. Leveau, and E. Ravelli, “Object coding of harmonic sound using sparse and structured representations,” in *Proc. of DAFx-07, Bordeaux, France*, 2007.

MODELING OF THE CARBON MICROPHONE NONLINEARITY FOR A VINTAGE TELEPHONE SOUND EFFECT

Sami Oksanen, Vesa Välimäki

Department of Signal Processing and Acoustics,
Aalto University, Espoo, Finland

sami.oksanen@aalto.fi, vesa.valimaki@tkk.fi

ABSTRACT

The telephone sound effect is widely used in music, television and the film industry. This paper presents a digital model of the carbon microphone nonlinearity which can be used to produce a vintage telephone sound effect. The model is constructed based on measurements taken from a real carbon microphone. The proposed model is a modified version of the sandwich model previously used for nonlinear telephone handset modeling. Each distortion component can be modeled individually based on the desired features. The computational efficiency can be increased by lumping the spectral processing of the individual distortion components together. The model incorporates a filtered noise source to model the self-induced noise generated by the carbon microphones. The model has also an input level depended noise generator for additional sound quality degradation. The proposed model can be used in various ways in the digital modeling of the vintage telephone sound.

1. INTRODUCTION

The age of telephone started during the 1870's when Bell patented the electromagnetic telephone. The viability of electroacoustic transmission was increased when Edison patented the carbon microphone. The transmission distances grew to a level where the telephone became an important method of communication due to the fact that they were easy to manufacture at a fairly low cost. Later on, carbon microphone performance was improved in various steps. About of century later, carbon microphones were outdated and replaced with more sophisticated microphone designs [1].

The telephony in the present day has changed greatly since early days. Traditional analog telephone networks were digitized, and later on cellular phones superseded local line telephones. Old fashioned analog telephone technology is still used in some third world countries, but is vanishing as the mobile communication technique is getting more popular. Some work has been done to maintain the cultural heritage of vintage audio recordings in form of digital modeling of recording and playback instruments of such era [2] and also in the field of vintage telephony [3].

The vintage telephone sound effect is widely used in the music industry. From a musical perspective, the telephone sound effect is typically used in modifying the singing voice. Complete music tracks can be processed with the telephone effect to make an illusion that they have been produced a long time ago. Another important field is the television and film industry where the telephone sound effect is widely used is occasions where telephone discussions are shown.

The scope of this paper is to introduce a method for modeling the nonlinear features of a carbon microphone to create a vintage telephone sound effect. The proposed model is flexible and can be adjusted to meet the demands of the computational efficiency. The modeling accuracy can be increased in occasions where more realistic outcome is wanted.

This paper is organized as follows. In Sec. 2 the key element of the vintage telephone, the carbon microphone, is reviewed. Next, the measured distortion characteristics of a carbon microphone are presented. In Sec. 3 the modeling of the carbon microphone nonlinearity is inspected. A novel method for the carbon microphone nonlinearity approximation is presented in Sec. 4 and a case example of modeling a carbon microphone using the proposed model is presented in Sec. 5. Finally, the conclusions are drawn in Sec. 6.

2. CARBON MICROPHONE MEASUREMENTS AND DISTORTION ANALYSIS

In general, telephone systems have a very limited transmission band and signal transmission is highly nonlinear [4]. Most of the nonlinearity originates in the carbon microphones. The second harmonic is usually the most dominant [5]. Transmission line terminations, telephone receivers and switches are also known to cause nonlinearities.

Carbon microphones are typically made of a metallic cup which is filled with carbon granules. On top of the cup there is a diaphragm which will transform the air pressure variation into a movement which will affect the resistance of the carbon granules. The carbon microphone is equipped with a DC bias voltage source which has one pole connected to the electrically connective diaphragm and the other to the bottom of the cup. The changing resistance generates electrical signal which can then be electrically transmitted. The harmonic frequency response of a carbon microphone is typically dominated by peaks around the 2 kHz and 3 kHz areas. Due to their construction, carbon microphones are known to produce harmonic distortion which gives the sound output its unique characteristics [5].

The nonlinear behavior of carbon microphones was analyzed using the nonlinear system identification method [6] based on a Hammerstein model [7]. The nonlinear system under investigation is excited with a swept-sine input signal while the system response is recorded. Then the system response is convolved with the time reversed excitation signal which is multiplied with an exponentially decaying amplitude envelope. As a result, a series of impulse responses representing the linear response (see Fig. 1 upper part) and corresponding harmonic distortion component responses is generated (see Fig. 1 lower part). The linear contribution is the

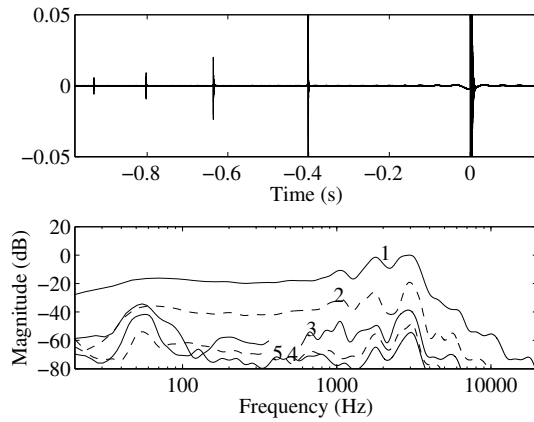


Figure 1: Upper figure, series of impulses resulting from the log sweep analysis of a carbon microphone. The impulse of a linear contribution is at the time origin and corresponding distortion component impulse responses are on the negative time axis. Lower figure, the linear response is marked with number 1 and the harmonic distortion components with numbers from 2 to 5 (third-octave smoothing is applied).

most dominant in the deconvolved impulse response and is preceded by the impulse responses of the distortion components.

An example of a carbon microphone response with a few low-level distortion components is presented in Fig 1. The linear response (marked with number 1) follows quite closely the single-button carbon microphone response described in [4]. The harmonic distortion components (marked with numbers 2-5) gives an approximate estimate of the total harmonic distortion as a function of frequency of the microphone under investigation. The second harmonic distortion component is the most dominant whereas the following components are more attenuated. The measurements were done for a telephone carbon button microphone from the late 60s.

The measurement setup consisted of a PC which was controlling the sample playback and recording. The measurement software was written in MATLAB [8]. The measurements were conducted in an anechoic chamber where the excitation signal was produced using an active loudspeaker with known frequency response. The reference signal was measured using B&K 4192 microphone. The carbon microphone under investigation was mounted to a LM Ericsson telephone handset from the 1970's. The carbon microphone was connected to a custom made microphone pre-amplifier which provided a 1.5-V bias voltage to the carbon microphone under investigation. The handset was mounted on a microphone stand that was placed at a 1 m distance in front of the speaker. The microphone responses were measured at approximately 80 dB SPL at the measurement point.

3. PROPOSED MODEL

The proposed model is based on the sandwich vintage telephone model presented by Välimäki *et al.* [9]. The nonlinearity of the proposed model is produced using the Chebyshev model for audio effects presented by Novak *et al.* [10]. The nonlinear components are modeled using Chebyshev polynomials together with

individual impulse responses for each distortion component. The main components of the proposed model are presented in Fig 2. First, the input signal is bandlimited by using a linear pre-filter (Line EQ). The nonlinearity is realized by using the output of the pre-filter as an input signal to the distortion component generation block where each distortion component is generated individually. Carbon microphones are known to generate noise [11, 12]. The self-induced noise of the carbon microphone is realized by adding a suitable amount of filtered noise to the pre-filtered and the distorted signal. The noise level is controlled with the G_p parameter. Some extra degradation to the output signal is created by adding some input signal dependent white noise to the filtered noise. Finally, the output signal is constructed by mixing the linear response with the distortion components and the generated noise. The system output is then fine-tuned with a band-limiting post filter (Post EQ). The nonlinear system modeling is done by processing each distortion component.

3.1. Pre-Filter (Line EQ)

The linear pre-filter is constructed based on the measured carbon microphone frequency response instead of the synthetically constructed response presented in [3]. The pre-filter can be realized by using the measured impulse response as FIR filter coefficients. A computationally more efficient realization can be achieved by using for example Prony's method to construct an IIR filter approximation of the impulse response.

3.2. Nonlinearity

According to Novak *et al.* [10] the nonlinearity is produced by reflecting the input signal to the corresponding harmonic distortion component frequency by using the first-order Chebyshev polynomial definition

$$T_n(x) = \cos(n\theta), \quad \text{where } x = \cos(\theta). \quad (1)$$

The corresponding Chebyshev polynomials $T_n(x)$ are defined recursively as

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad n = 2, 3, \dots, \quad (2)$$

using the initial conditions

$$T_0(x) = 1, \quad T_1(x) = x. \quad (3)$$

In the proposed model the nonlinearity is produced by using the linear response for each distortion component. However, this is not the optimal solution for modeling of the exact physical behavior of the carbon microphone, but it offers a computationally more efficient solution compared to the use of individual impulse responses for each component.

To add more realism to the model, the distortion component can be adjusted by using a separate Regalia-Mitra equalization filter [13]. This equalization filter can be used to fine-tune the computationally efficient distortion components realization, to add desired spectral tilt or to boost or attenuate certain frequency range. The transfer function of the second-order peak filter is given by

$$H(z) = 1 + \frac{H_0}{2}[1 - A_2(z)], \quad (4)$$

where A_2 is realized using a second-order allpass filter section. The output of the nonlinear section is constructed by creating a weighted sum of the distortion components using the gain coefficients $G_2 \dots G_n$ as shown in Fig 2.

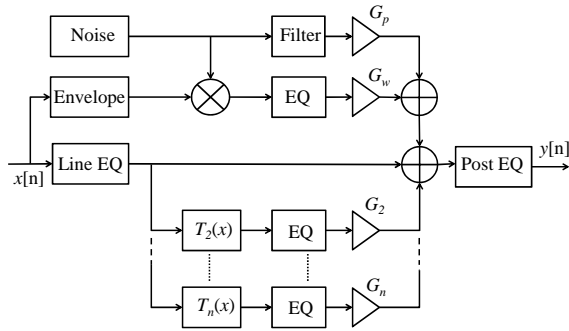


Figure 2: Block diagram of the proposed effect model.

3.3. Self-Induced Noise Generation

The input signal dependent noise can be used to increase the impression of failures in the microphone signal. The input signal dependency is created by modulating the white noise amplitude with the input signal envelope. The input signal envelope is obtained by applying the Hilbert transform to it and lowpass filtering the result with a second-order Butterworth filter with cutoff frequency set to 700 Hz. From the measurement results can be seen that the microphone output produces a background noise spectrum close to a $1/f$ relation. The constant background noise is realized by adding a suitable amount of filtered noise to the distorted signal.

3.4. Post-Filter (Post EQ)

The post-filter is used to reduce the bandwidth to the desired level. In telephony the transmission band is typically from 300 Hz to 3400 Hz. A post-filter meeting this requirement can be implemented by using a fourth-order Butterworth bandpass filter with 400 Hz and 2800 Hz as transition points. The sound effect can be enriched by setting the upper threshold value even higher. This would result in having more harmonic distortion components in the sound effect output. The steepness of the band limitation can also be adjusted to meet the desired output. Finally the distortion components are summed with the linear response and the noise output to generate the sound effect output.

4. MODELING OF THE CARBON MICROPHONE NONLINEARITY

A digital implementation of the telephone sound effect may be based on the sandwich model where a static nonlinearity is preceded by a linear pre-filter and followed by a linear bandlimiting post-filter (See Fig. 3). Originally this model was used to estimate the nonlinear behavior of telephone handsets by Quatieri *et al.* [14]. Recently, this method was applied to telephone sound modeling by Välimäki *et al.* [9].

The nonlinearity approximation is based on Taylor series polynomials which are used as static nonlinearity functions [14]. The following approximation is controlled with the nonlinearity parameter α and is valid for low values

$$Q(u) = \begin{cases} (1 - \alpha)u \sum_{k=1}^{\infty} \alpha^k u^k, & u \leq 1, |\alpha| < 1 \\ 1, & u > 1, \end{cases} \quad (5)$$

where u is the input signal and k is the model order. An example of a logsweep analysis of this model output using a measured mi-

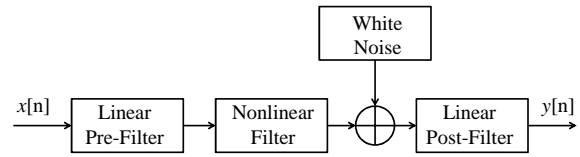


Figure 3: Sandwich model for the telephone sound effect where the nonlinearity is produced with a static nonlinear function [9].

crophone response is presented in Fig. 4. The model is capable of producing the linear response but fails in producing accurately the distortion components. The level of third-order and higher harmonic components is greatly attenuated compared to the measured response.

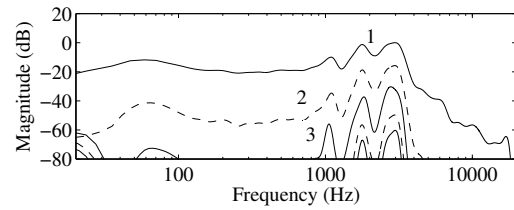


Figure 4: Logsweep analysis of the simple sandwich model with a static nonlinearity ($N=5$). The linear response is marked with number 1 and the harmonic distortion components with numbers from 2 to 5 (third-octave smoothing is applied).

5. RESULTS

The model was tested by constructing the pre-filter from the measured carbon microphone response and the nonlinearity was modeled based on the harmonic distortion analysis. The FIR pre-filter was constructed from the measured system impulse response. To limit the model computational complexity the distortion components were composed from the pre-filtered input signal. This does not exactly follow the actual carbon microphone distortion component behavior, but results in more realistic distortion characteristics (Fig. 5) compared to the simple sandwich model (Fig. 4).

The pre-filtered signal was conditioned to correct the low-frequency attenuation by designing a second-order Regalia-Mitra equalization filter by boosting the filter resonance frequency of 300 Hz by 30 dB. The resonance width was controlled with the Q-value of 0.4. The resulting logsweep analysis of the model output is presented in Fig. 5. It can be seen that the distortion components have a larger gain at the frequencies from 100 Hz to 1 kHz compared to the simplified model. The magnitude of each distortion component can be adjusted by setting the gain coefficients ($G_2 \dots G_n$) to match the desired level. In this example the gain coefficients were adjusted as follows, $G_2=0.7$, $G_3=0.5$, $G_4=0.4$, and $G_5=0.5$ as the linear part gain was 1. The background noise was controlled with the filtered noise gain setting of $G_p=0.009$.

The model output for a 1-kHz sinusoidal excitation signal and corresponding measurement are presented in Fig 6. It can be seen that the distortion components produced with this model follow quite closely the measured values. However, it should be noted that the distortion component spectra are approximated with the

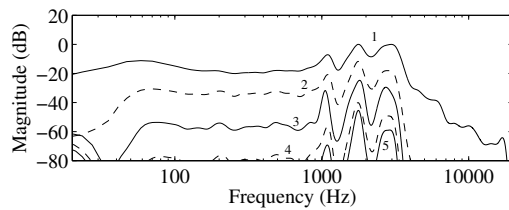


Figure 5: Logsweep analysis of the model output. The linear response is marked with number 1 and the harmonic distortion components with numbers from 2 to 5 (third-octave smoothing is applied). compare with lower part of the Fig. 1.

measured linear microphone response and this might result in deviation between the model output compared to the actual response to be modeled. The effect can be reduced by using the measured responses in each modeled distortion component. Sound examples are made available through a website for downloading¹.

6. CONCLUSIONS

A method for modeling carbon microphone nonlinearities for the telephone sound effect was presented in this paper. The proposed model consists of the modified sandwich nonlinear model and noise generation block. The nonlinearity is modeled by using Chebyshev polynomials and distortion component frequency responses individually for each distortion component. The model allows to control each distortion component and the model can be scaled according to computational limitations.

A case study of modeling carbon button microphone nonlinearity was presented and was found out that the proposed model can be used to approximate the nonlinearities in telephone sound effect. In this example the computational efficiency was improved by modeling the distortion components using the linear frequency response instead of individual responses. The input signal dependent noise can also be added to the degradation process to create more disturbances to the sound.

7. ACKNOWLEDGMENTS

The authors would like to thank Dr. Jyri Pakarinen for the assistance provided during the distortion analysis. The work of Sami Oksanen is funded by the Academy of Finland, project no. 122815.

8. REFERENCES

- [1] J. K. Hilliard, "A review of early developments in electroacoustics in the U.S.A.," in *Proc. 57th Audio Engineering Society Convention*, May 1977.
- [2] V. Välimäki, S. González, O. Kimmelma, and J. Parviainen, "Digital audio antiaging – signal processing methods for imitating the sound quality of historical recordings," *J. Audio Eng. Soc.*, vol. 56, no. 3, pp. 115–139, 2008.
- [3] S. Oksanen and V. Välimäki, "Digital modeling of the vintage telephone sound," in *Proc. ICMC 2011*, Huddersfield, UK, Aug. 2011.
- [4] H. F. Olson, *Acoustical Engineering*, Professional Audio Journals, Inc., 1991.
- [5] M. T. Abuelma'atti, "Harmonic and intermodulation distortion in carbon microphones," *Applied Acoustics*, vol. 31, no. 4, pp. 233–243, 1990.
- [6] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine signal," in *Proc. AES 108th Convention*, Paris, Feb. 2000.
- [7] M. Rébillat, R. Hennequin, E. Corteel, and B. Katz, "Identification of cascade of Hammerstein models for the description of nonlinearities in vibrating devices," *J. Sound and Vibration*, vol. 330, no. 5, pp. 1018 – 1038, 2011.
- [8] J. Pakarinen, "Distortion analysis toolkit - a software tool for easy analysis of nonlinear audio systems," *EURASIP J. Advances in Signal Processing, Special Issue on Digital Audio Effects*, 2010, Article ID 617325.
- [9] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, "Virtual analog effects," in *DAFx Digital Audio Effects*, Udo Zölzer, Ed., pp. 473–522. 2011.
- [10] A. Novak, L. Simon, P. Lotton, and J. Gilbert, "Chebyshev model and synchronized swept sine method in nonlinear audio effect modeling," in *Proc. DAFx-10*, Graz, Austria, Sept. 6–10, 2010.
- [11] E. Schiller, "Self induced noise in carbon microphones," in *Proc. 17th AES Convention*, October 1965.
- [12] J. R. Sank, "Microphones," *J. Audio Eng. Soc.*, vol. 33, no. 7/8, pp. 514–547, 1985.
- [13] P. Regalia and S. Mitra, "Tunable digital frequency response equalization filters," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 35, no. 1, pp. 118–120, January 1987.
- [14] T.F. Quatieri, D.A. Reynolds, and G.C. O'Leary, "Estimation of handset nonlinearity with application to speaker recognition," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 5, pp. 567–584, Sept. 2000.

¹<http://www.acoustics.hut.fi/go/dafx11-carbon>

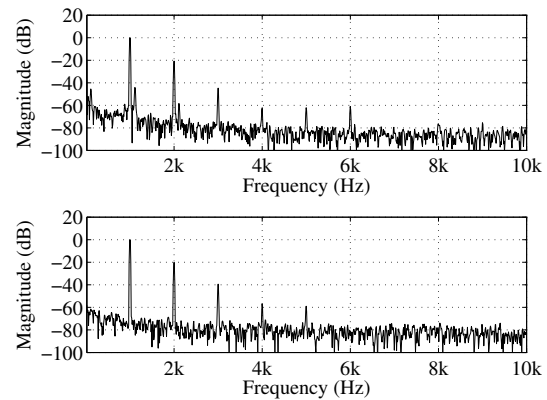


Figure 6: Response to a 1-kHz sinusoidal signal, measured carbon microphone response in lower figure and model output in lower figure.

PHYSICAL MODELLING OF A WAH-WAH EFFECT PEDAL AS A CASE STUDY FOR APPLICATION OF THE NODAL DK METHOD TO CIRCUITS WITH VARIABLE PARTS

Martin Holters, Udo Zölzer

Department of Signal Processing and Communications

Helmut Schmidt University

Hamburg, Germany

`martin.holters@hsu-hh.de, udo.zoelzer@hsu-hh.de`

ABSTRACT

The nodal DK method is a systematic way to derive a non-linear state-space system as a physical model for an electrical circuit. Unfortunately, calculating the system coefficients requires inversion of a relatively large matrix. This becomes a problem when the system changes over time, requiring continuous recomputation of the coefficients. In this paper, we present an extension of the DK method to more efficiently handle variable circuit elements. The method is exemplified with the Dunlop Crybaby wah-wah effect pedal, as the continuous change of the potentiometer position is an extremely important aspect of the wah-wah effect.

1. INTRODUCTION

The nodal DK method is a systematic way to derive a non-linear state-space system as a physical model for an electrical circuit [1, 2]. Computation of the system coefficients involves inverting a system matrix, the size of which is determined by the number of circuit nodes, which is typically significantly higher than the order of the resulting system. While for time-invariant systems the inversion has to be performed only once, problems arise when variable elements like potentiometers are introduced. In that case, the computational load for continuous recalculation of the coefficients may become prohibitively large when following the straight-forward approach. Therefore, in this paper we present a more efficient recomputation scheme.

We exemplify the method with the Dunlop Crybaby wah-wah effect pedal, as the continuous change of the potentiometer position is an extremely important aspect of the wah-wah effect. Whether a full-featured non-linear physical model of the wah-wah is really necessary or simpler approaches (e.g. as described in [3]) suffice is anyone's choice; but the circuit is well suited to explain the technique presented in this paper which is why it is used here. The main part of the circuit is depicted in Figure 1. Omitted here are the input stage and the power-supply filter. The input stage is a simple AC-coupled emitter-follower which for reasonable input amplitudes may be well approximated by a linear high-pass filter. The output of that filter is used as the input voltage V_i . The supply voltage V_{cc} is obtained from the battery by an RC network which stabilizes the supply to about 90% of the battery voltage. The stabilization is sufficient to assume constant V_{cc} .

2. REVIEW OF THE DK METHOD

We shall start with a review of the nodal DK method because compared to [1], we use a slightly different discretization scheme

for inductors and we need to more rigorously define the involved steps in terms of matrix operations in order to derive the simplified update scheme.

2.1. Companion circuits

The first step is to replace the energy-storing elements (capacitors and inductors) with so called companion circuits, obtained from discretization. In particular, from the differential equations

$$i_C = C \frac{d}{dt} v_C \quad v_L = L \frac{d}{dt} i_L \quad (1)$$

for capacitor and inductor, by applying the trapezoidal discretization scheme, we get the discrete-time approximations

$$\frac{1}{2} (i_C(n) + i_C(n-1)) = \frac{C}{T} (v_C(n) - v_C(n-1)) \quad (2)$$

$$\frac{1}{2} (v_L(n) + v_L(n-1)) = \frac{L}{T} (i_L(n) - i_L(n-1)) \quad (3)$$

where T denotes the sampling interval. Solving for the current time step's currents then yields

$$i_C(n) = \frac{2C}{T} (v_C(n) - v_C(n-1)) - i_C(n-1) \quad (4)$$

$$i_L(n) = \frac{T}{2L} (v_L(n) + v_L(n-1)) + i_L(n-1), \quad (5)$$

where both the voltages and the currents hold state information. We may, however, introduce canonical states $x_C(n)$ and $x_L(n)$ by substituting

$$i_C(n) = x_C(n) - \frac{2C}{T} v_C(n) \quad (6)$$

$$i_L(n) = -x_L(n) - \frac{T}{2L} v_L(n) \quad (7)$$

in Equations (4) and (5) to get

$$x_C(n) - \frac{2C}{T} v_C(n) = \frac{2C}{T} v_C(n) - x_C(n-1) \quad (8)$$

$$-x_L(n) - \frac{T}{2L} v_L(n) = \frac{T}{2L} v_L(n) - x_L(n-1), \quad (9)$$

leading to the final state update equations

$$x_C(n) = 2 \frac{2C}{T} v_C(n) - x_C(n-1) \quad (10)$$

$$x_L(n) = -2 \frac{T}{2L} v_L(n) + x_L(n-1). \quad (11)$$

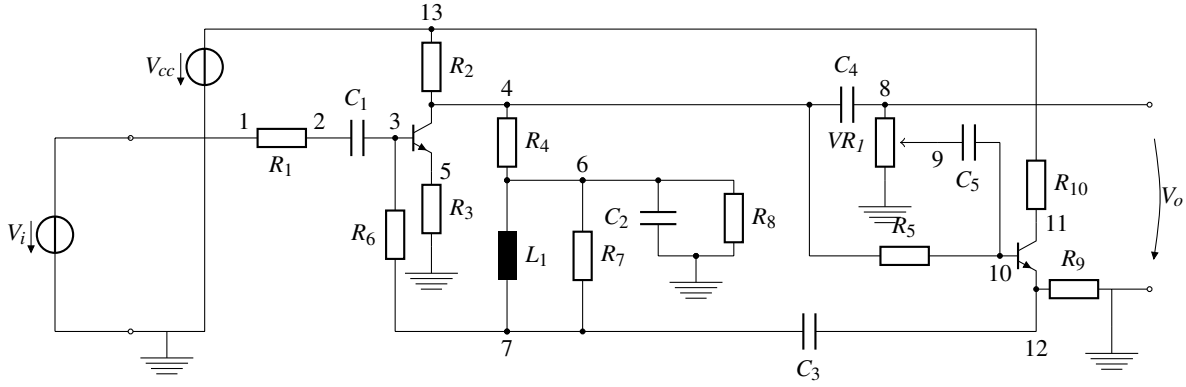
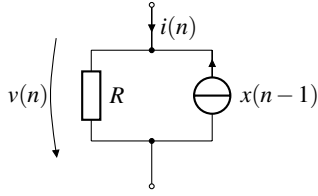


Figure 1: Schematic of the analyzed Crybaby circuit with numbering of the nodes indicated.


 Figure 2: Companion circuit for capacitor ($R = \frac{T}{2C}$) and inductor ($R = \frac{2L}{T}$).

By substituting Equations (10) and (11) in Equations (6) and (7), we obtain

$$i_C(n) = \frac{2C}{T} v_C(n) - x_C(n-1) \quad (12)$$

$$i_L(n) = \frac{T}{2L} v_L(n) - x_L(n-1), \quad (13)$$

defining our companion circuit as shown in Figure 2. We hence replace all energy-storing elements with resistors and current sources, where the current sources depend on the previous time step and thereby hold the state information. We can thus analyze a circuit containing only resistors and sources (as described next) and then use the results to update the states (by Equations (10) and (11)), resulting in the next steps source currents.

2.2. Nodal K-method

The circuit is now analyzed using the nodal K-method. We define a reference node, the ground node as is common, and introduce the potentials φ_m of the other nodes and the currents $i_{s,i}$ through the voltage sources as unknowns. We now apply the Kirchhoff current law at all nodes except for the reference node. For example, for the circuit of Figure 1, the first three nodes yield the equations

$$\frac{1}{R_1}(\varphi_1 - \varphi_2) + i_{s,1} = 0 \quad (14)$$

$$\frac{1}{R_1}(\varphi_2 - \varphi_1) + \frac{1}{R_{C_1}}(\varphi_2 - \varphi_3) = x_{C_1} \quad (15)$$

$$\frac{1}{R_6}(\varphi_3 - \varphi_7) + \frac{1}{R_{C_1}}(\varphi_3 - \varphi_2) = -x_{C_1} - i_{B1} \quad (16)$$

where the current source of the companion circuit of C_1 is assumed to be pointing to the left. The transistor currents (the base current i_{B1} in the example equations) are for the moment assumed to be known and introduced as additional current sources.

The complete system may be written as

$$(N_R^T G_R N_R + N_v^T R_v^{-1} N_v + N_x^T G_x N_x) \varphi + N_u^T i_s = N_x^T x + N_n^T i_n \quad (17)$$

where N_R , N_v , N_x , N_u and N_n are oriented incidence matrices which specify the nodes to which the resistors, potentiometers, energy-storing elements, voltage sources and non-linear elements, respectively, are connected,

$$G_R = \text{diag} \left(\frac{1}{R_1}, \dots, \frac{1}{R_{10}} \right) \quad (18)$$

is a diagonal matrix with the reciprocal resistances,

$$R_v = \text{diag}(\alpha VR_I, (1 - \alpha)VR_I) \quad (19)$$

is a diagonal matrix with the variable resistances parameterized with the potentiometer position α ,

$$G_x = \text{diag} \left(\frac{2C_1}{T}, \dots, \frac{2C_5}{T}, \frac{T}{2L_1} \right) \quad (20)$$

is a diagonal matrix with the reciprocal resistances of the companion circuits, φ the vector of unknown node potentials, i_s the vector of unknown voltage source currents, x the current states of the companion circuits and i_n the currents of the non-linear elements.

The incidence matrices $N_{(\cdot)}$ contain one row per circuit element and one column per node (except the reference node). The entries are mostly zero; at most two entries per element are non-zero: a 1 in the column of the node where the positive pole of the element is connected, a -1 where the negative pole is connected; connections to the reference node are omitted. Polarity of the sources is obvious, polarity of the passive elements may be chosen

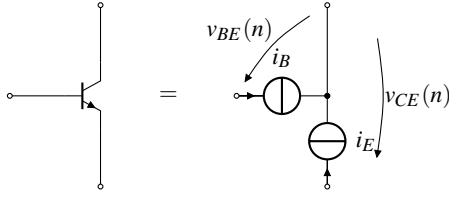


Figure 3: Equivalent circuit element used for the transistors.

at will. For the system of Figure 1, we find

$$N_R = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix} \quad (21)$$

$$N_v = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (22)$$

$$N_x = \begin{pmatrix} 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (23)$$

$$N_u = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (24)$$

$$N_n = \begin{pmatrix} 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \end{pmatrix} \quad (25)$$

where we describe the transistors in terms of the base and emitter currents as shown in Figure 3.

To fully describe the circuit, we furthermore need to exploit that the voltage sources directly give relations among the node potentials, in our case

$$\varphi_1 = V_i \quad (26)$$

$$\varphi_{13} = V_{cc} \quad (27)$$

which may be combined with Equation (17) to give

$$S \begin{pmatrix} \varphi \\ i_s \end{pmatrix} = \begin{pmatrix} N_x^T \\ \mathbf{0} \end{pmatrix} x + \begin{pmatrix} \mathbf{0} \\ I \end{pmatrix} u + \begin{pmatrix} N_n^T \\ \mathbf{0} \end{pmatrix} i_n \quad (28)$$

where $u = (V_i \ V_{cc})^T$ is the source vector, $\mathbf{0}$ and I are the all-zero and identity matrix with size as required by context, and

$$S = \begin{pmatrix} N_R^T G_R N_R + N_v^T R_v^{-1} N_v + N_x^T G_x N_x & N_u^T \\ N_u & \mathbf{0} \end{pmatrix}. \quad (29)$$

By left-multiplying with S^{-1} and the respective incidence matrices,

we can then extract all required voltages by

$$v_x = (N_x \ \mathbf{0}) S^{-1} \left(\begin{pmatrix} N_x^T \\ \mathbf{0} \end{pmatrix} x + \begin{pmatrix} \mathbf{0} \\ I \end{pmatrix} u + \begin{pmatrix} N_n^T \\ \mathbf{0} \end{pmatrix} i_n \right) \quad (30)$$

$$v_n = (N_n \ \mathbf{0}) S^{-1} \left(\begin{pmatrix} N_x^T \\ \mathbf{0} \end{pmatrix} x + \begin{pmatrix} \mathbf{0} \\ I \end{pmatrix} u + \begin{pmatrix} N_n^T \\ \mathbf{0} \end{pmatrix} i_n \right) \quad (31)$$

$$v_o = (N_o \ \mathbf{0}) S^{-1} \left(\begin{pmatrix} N_x^T \\ \mathbf{0} \end{pmatrix} x + \begin{pmatrix} \mathbf{0} \\ I \end{pmatrix} u + \begin{pmatrix} N_n^T \\ \mathbf{0} \end{pmatrix} i_n \right) \quad (32)$$

where v_x , v_n and v_o are the voltages across the energy-storing elements, the non-linear elements and the desired output voltage, respectively. The incidence matrix N_o determines the output voltage and in the example is given by

$$N_o = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (33)$$

2.3. Non-linear state-space system

We are now ready to formulate the non-linear state space system. We first rewrite Equations (10) and (11) for a complete system as

$$x(n) = Z \cdot (2G_x v_x(n) - x(n-1)) \quad (34)$$

where

$$Z = \text{diag}(1 \ 1 \ 1 \ 1 \ 1 \ -1) \quad (35)$$

is a diagonal matrix which contains a 1 for a capacitor and a -1 for an inductor to take care of the differing signs in Equation (10) and Equation (11). Substituting Equation (30), we obtain the state update equation

$$x(n) = Ax(n-1) + Bu(n) + Ci_n(n) \quad (36)$$

with

$$A = 2ZG_x(N_x \ \mathbf{0}) S^{-1}(N_x \ \mathbf{0})^T - Z \quad (37)$$

$$B = 2ZG_x(N_x \ \mathbf{0}) S^{-1}(\mathbf{0} \ I)^T \quad (38)$$

$$C = 2ZG_x(N_x \ \mathbf{0}) S^{-1}(N_n \ \mathbf{0})^T. \quad (39)$$

Similarly, we may rewrite Equation (32) to obtain the output equation

$$y(n) = v_o(n) = Dx(n-1) + Eu(n) + Fi_n(n) \quad (40)$$

with

$$D = (N_o \ \mathbf{0}) S^{-1}(N_x \ \mathbf{0})^T \quad (41)$$

$$E = (N_o \ \mathbf{0}) S^{-1}(\mathbf{0} \ I)^T \quad (42)$$

$$F = (N_o \ \mathbf{0}) S^{-1}(N_n \ \mathbf{0})^T. \quad (43)$$

Finally, we obtain the voltages across the non-linear elements by rewriting Equation (31) to

$$v_n(n) = Gx(n-1) + Hu(n) + Ki_n(n) \quad (44)$$

with

$$G = (N_n \ \mathbf{0}) S^{-1}(N_x \ \mathbf{0})^T \quad (45)$$

$$H = (N_n \ \mathbf{0}) S^{-1}(\mathbf{0} \ I)^T \quad (46)$$

$$K = (N_n \ \mathbf{0}) S^{-1}(N_n \ \mathbf{0})^T. \quad (47)$$

Note that Equation (44) defines a relationship between the voltages across and the currents through the non-linear circuit elements due to the external circuit and its state. Additionally, the non-linear elements define such a relationship by themselves; combining these two will allow to solve for the currents $i_n(n)$ or equivalently the voltages $v_n(n)$.

In the example, the non-linear elements are transistors which will be modelled using the Ebers-Moll-equations. In particular, we get

$$i_{n,1} = \frac{I_S}{\beta_F} \left(e^{\frac{v_{n,2}-v_{n,1}}{V_T}} - 1 \right) + \frac{I_S}{\beta_R} \left(e^{\frac{-v_{n,1}}{V_T}} - 1 \right) \quad (48)$$

$$i_{n,2} = -I_S \left(e^{\frac{v_{n,2}-v_{n,1}}{V_T}} - 1 \right) + \frac{I_S(\beta_R - 1)}{\beta_R} \left(e^{\frac{-v_{n,1}}{V_T}} - 1 \right) \quad (49)$$

$$i_{n,3} = \frac{I_S}{\beta_F} \left(e^{\frac{v_{n,4}-v_{n,3}}{V_T}} - 1 \right) + \frac{I_S}{\beta_R} \left(e^{\frac{-v_{n,3}}{V_T}} - 1 \right) \quad (50)$$

$$i_{n,4} = -I_S \left(e^{\frac{v_{n,4}-v_{n,3}}{V_T}} - 1 \right) + \frac{I_S(\beta_R - 1)}{\beta_R} \left(e^{\frac{-v_{n,3}}{V_T}} - 1 \right). \quad (51)$$

The processing for time step n now proceeds according to the following schedule:

1. Calculate $\mathbf{p}(n) = \mathbf{G}\mathbf{x}(n-1) + \mathbf{H}\mathbf{u}(n)$.
2. Numerically solve $\mathbf{p}(n) + \mathbf{K}\mathbf{i}_n(n) - \mathbf{v}_n(n) = 0$ together with Equations (48) to (51) to obtain $\mathbf{i}_n(n)$, e.g. using Newton iteration.
3. Compute the output with Equation (40).
4. Perform the state update with Equation (36).

3. EFFICIENT HANDLING OF VARIABLE ELEMENTS

As all of the matrices defined above can be precomputed, the main computational burden lies in solving the non-linear equation. However, in case of a variable element, like the potentiometer VR_I , the system matrices need to be recomputed every time the element changes—and for the wah-wah effect, the potentiometer will change almost continuously. At first glance, it looks like this means the matrix \mathbf{S} has to be inverted regularly, which would be rather unfortunate, as inverting a 15×15 matrix would mean a significant additional computational load.

Fortunately, we may decompose the system matrix as

$$\mathbf{S} = \mathbf{S}_0 + (\mathbf{N}_v \quad \mathbf{0})^T \mathbf{R}_v^{-1} (\mathbf{N}_v \quad \mathbf{0}) \quad (52)$$

where

$$\mathbf{S}_0 = \begin{pmatrix} \mathbf{N}_R^T \mathbf{G}_R \mathbf{N}_R + \mathbf{N}_x^T \mathbf{G}_x \mathbf{N}_x & \mathbf{N}_u^T \\ \mathbf{N}_u & \mathbf{0} \end{pmatrix} \quad (53)$$

is independent of the potentiometer setting. We may now use the Woodbury identity [4] to rewrite the inverse as

$$\mathbf{S}^{-1} = \mathbf{S}_0^{-1} - \mathbf{S}_0^{-1} (\mathbf{N}_v \quad \mathbf{0})^T (\mathbf{R}_v + \mathbf{Q})^{-1} (\mathbf{N}_v \quad \mathbf{0}) \mathbf{S}_0^{-1} \quad (54)$$

with

$$\mathbf{Q} = (\mathbf{N}_v \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_v \quad \mathbf{0})^T \quad (55)$$

which reduces the size of the matrix to be regularly inverted to 2×2 , as \mathbf{S}_0^{-1} may be precomputed off-line. As an additional benefit, we do not have to worry about inverting \mathbf{R}_v , which would require special attention for $\alpha = 0$ and $\alpha = 1$ when the resistances become

zero. Plugging Equation (54) into the definition of the state-space system matrices, we obtain

$$\mathbf{A} = \mathbf{A}_0 - 2\mathbf{Z}\mathbf{G}_x\mathbf{U}_x(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_x^T \quad (56)$$

$$\mathbf{B} = \mathbf{B}_0 - 2\mathbf{Z}\mathbf{G}_x\mathbf{U}_x(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_u^T \quad (57)$$

$$\mathbf{C} = \mathbf{C}_0 - 2\mathbf{Z}\mathbf{G}_x\mathbf{U}_x(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_n^T \quad (58)$$

$$\mathbf{D} = \mathbf{D}_0 - \mathbf{U}_o(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_x^T \quad (59)$$

$$\mathbf{E} = \mathbf{E}_0 - \mathbf{U}_o(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_u^T \quad (60)$$

$$\mathbf{F} = \mathbf{F}_0 - \mathbf{U}_o(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_n^T \quad (61)$$

$$\mathbf{G} = \mathbf{G}_0 - \mathbf{U}_n(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_x^T \quad (62)$$

$$\mathbf{H} = \mathbf{H}_0 - \mathbf{U}_n(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_u^T \quad (63)$$

$$\mathbf{K} = \mathbf{K}_0 - \mathbf{U}_n(\mathbf{R}_v + \mathbf{Q})^{-1}\mathbf{U}_n^T \quad (64)$$

with the constant matrices

$$\mathbf{U}_x = (\mathbf{N}_x \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_v \quad \mathbf{0})^T \quad (65)$$

$$\mathbf{U}_o = (\mathbf{N}_o \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_v \quad \mathbf{0})^T \quad (66)$$

$$\mathbf{U}_n = (\mathbf{N}_n \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_v \quad \mathbf{0})^T \quad (67)$$

$$\mathbf{U}_u = (\mathbf{0} \quad \mathbf{I}) \mathbf{S}_0^{-1} (\mathbf{N}_v \quad \mathbf{0})^T \quad (68)$$

$$\mathbf{A}_0 = 2\mathbf{Z}\mathbf{G}_x(\mathbf{N}_x \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_x \quad \mathbf{0})^T - \mathbf{Z} \quad (69)$$

$$\mathbf{B}_0 = 2\mathbf{Z}\mathbf{G}_x(\mathbf{N}_x \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{0} \quad \mathbf{I})^T \quad (70)$$

$$\mathbf{C}_0 = 2\mathbf{Z}\mathbf{G}_x(\mathbf{N}_x \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_n \quad \mathbf{0})^T \quad (71)$$

$$\mathbf{D}_0 = (\mathbf{N}_o \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_x \quad \mathbf{0})^T \quad (72)$$

$$\mathbf{E}_0 = (\mathbf{N}_o \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{0} \quad \mathbf{I})^T \quad (73)$$

$$\mathbf{F}_0 = (\mathbf{N}_o \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_n \quad \mathbf{0})^T \quad (74)$$

$$\mathbf{G}_0 = (\mathbf{N}_n \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_x \quad \mathbf{0})^T \quad (75)$$

$$\mathbf{H}_0 = (\mathbf{N}_n \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{0} \quad \mathbf{I})^T \quad (76)$$

$$\mathbf{K}_0 = (\mathbf{N}_n \quad \mathbf{0}) \mathbf{S}_0^{-1} (\mathbf{N}_n \quad \mathbf{0})^T. \quad (77)$$

Note that all matrices involved in Equations (56) to (64) are relatively small compared to the system matrix \mathbf{S} , allowing efficient recomputation whenever the potentiometer setting changes.

4. RESULTS

The above model, supplemented with a simple first-order high-pass for the AC-coupled input buffer omitted from the above circuit analysis, has been implemented as an LV2¹ plugin. The matrix operations make use of the GNU Scientific Library². The non-linear equation is solved with a simple damped Newton iteration, where the solution of the previous time-step is used as starting point. This allows convergence in relatively few (typically less than ten) iterations, making real-time operation even on not quite up-to-date PC hardware possible. The plugin along with audio examples may be found at <http://ant.hsu-hh.de/dafox2011/wahwah>.

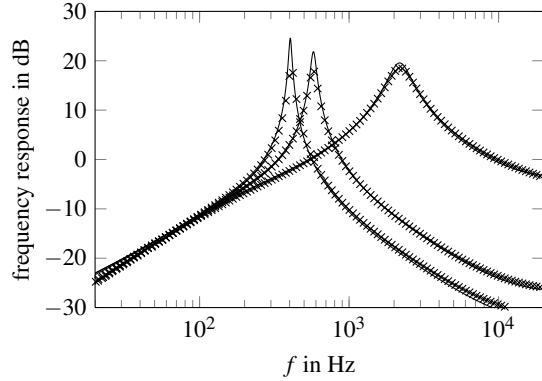
The model has been tested using the nominal component values and parameters given in Table 1. As a first test, the small-signal frequency responses of the model and the real circuit are compared

¹See <http://lv2plug.in/>.

²See <http://www.gnu.org/software/gsl/>.

Table 1: Component values and parameters used in the model.

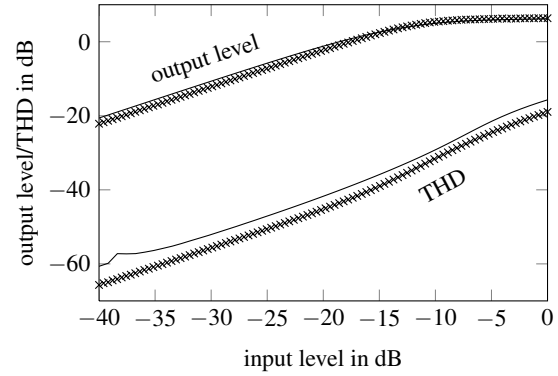
R_1	68	k Ω	C_1	10	nF
R_2	22	k Ω	C_2	4.7	μ F
R_3	390	Ω	C_3	10	nF
R_4	470	k Ω	C_4	220	nF
R_5	470	k Ω	C_5	220	nF
R_6	1.5	k Ω	L_1	500	mH
R_7	33	k Ω	V_{cc}	8.15	V
R_8	82	k Ω	V_t	26	mV
R_9	10	k Ω	I_s	20.3	fA
R_{10}	1	k Ω	β_f	1430	
VR_I	100	k Ω	β_r	4	

Figure 4: Comparison of measured (crosses) and simulated (line) small-signal frequency responses for $\alpha = 0.008, 0.570$, and 0.999 .

for three different potentiometer settings³. The corresponding value of α was determined by measuring the potentiometer. The resulting frequency responses are shown in Figure 4. Clearly, the model fits the real circuit quite well. Any deviations are easily explained by the tolerances of the physical circuit elements; especially the slightly different gain at the peak is no surprise, as the relative high Q-factor makes the circuit very sensitive to component tolerances.

The second experiment conducted is concerned with the non-linear behavior. Even for moderate input levels, the high gain at the peak frequency may result in a clipped output signal. As a very simple form of analysis, Figure 5 depicts the output level and total harmonic distortion (THD) in dependence on the input level for a sinusoidal input. The frequency of the sinusoid is chosen to be at the peak of the small-signal frequency response. As expected, for small input amplitudes, the simulated output has slightly higher output level corresponding to the higher small-signal gain at the peak frequency. For higher input amplitudes, both the real circuit and the simulation exhibit a smooth saturation when the output level approaches 6 dB. This smooth saturation is also resembled by the THD curve, which shows a steady increase for both measurement and simulation and stays at moderate levels even for a 0 dB input, where the overall gain is reduced by about 13 dB compared to the small-signal gain. While the THD of the simulation is constantly a few dB higher than that of the real circuit, the shape of the curves is very similar and it may be expected that better matching of the component values and the transistor parameters would bring the

³Note that the extremal positions cannot be reached due to mechanical restrictions when the potentiometer is mounted in the enclosure.

Figure 5: Comparison of measured (crosses) and simulated (line) output RMS level and THD in dependence on input RMS level for a sinusoid of 719 Hz with potentiometer setting $\alpha = 0.770$.

curves to close fit.

5. CONCLUSION

We have reviewed the nodal DK-method with a focus on efficient handling of variable circuit components, i.e. potentiometers. To that end, we have given a formulation of the DK-method with explicit use of the incidence matrices and diagonal matrices holding the component values, from which a system matrix is constructed that has to be inverted to obtain the coefficient matrices of a non-linear state-space model. This explicit form allows efficient handling of changing component values when the number of variable parts is low compared to the total number of parts. In that case, changing the variable component values results in a low-rank update of the system matrix, and the required inversion may be carried out efficiently using the Woodbury identity.

The method was presented by example of the Crybaby wah-wah circuit. The derived model provides a reasonable approximation of the real circuit considering that the nominal component values were used and no parameter matching to the real circuit was performed. The model was implemented as an LV2 plugin which proved the real-time capability of the model even on somewhat outdated PC hardware.

6. REFERENCES

- [1] D.T. Yeh, J.S. Abel, and J.O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—Part I: Theoretical development," *IEEE Trans. Audio, Speech, and Language Process.*, vol. 18, no. 4, pp. 728–737, May 2010.
- [2] D.T. Yeh, *Digital Implementation of Musical Distortion Circuits by Analysis and Simulation*, Ph.D. thesis, Stanford University, 2009.
- [3] J.O. Smith, *Making Virtual Electric Guitars and Associated Effects Using Faust*, chapter "Adding a Wah Pedal", Online: https://ccrma.stanford.edu/realsimple/faust_strings/Adding_Wah_Pedal.html.
- [4] K.B. Petersen and M.S. Pedersen, "The matrix cookbook," Oct. 2008, Version 20081110.

AUTOMATED CALIBRATION OF A PARAMETRIC SPRING REVERB MODEL

Hannes Gamper,^{*}

Department of Media Technology
Aalto University
Espoo, Finland

first[dot]last[at]tml.hut.fi

Julian Parker and Vesa Välimäki,[†]

Department of Signal Processing and Acoustics,
Aalto University
Espoo, Finland

first[dot]last[at]tkk.fi

ABSTRACT

The calibration of a digital spring reverberator model is crucial for the authenticity and quality of the sound produced by the model. In this paper, an automated calibration of the model parameters is proposed, by analysing the spectrogram, the energy decay curve, the spectrum, and the autocorrelation of the time signal and spectrogram. A visual inspection of the spectrograms as well as a comparison of sound samples proves the approach to be successful for estimating the parameters of reverberators with one, two and three springs. This indicates that the proposed method is a viable alternative to manual calibration of spring reverberator models.

1. INTRODUCTION

Spring reverberation is an early method of artificial reverberation, introduced by Laurens Hammond in the 1940s [1]. Its small size and low cost compared to the contemporary methods of artificial reverberation at the time, such as plates or chambers, led to its wide use both in studio applications and within electrical musical instruments and amplifiers. The special sound of the spring reverberator, caused by the highly dispersive nature of wave propagation on the spring, became valued as a musical effect distinct from standard reverberation.

A spring reverberator consists of one or more helical metal springs, connected in parallel, series or in a hybrid configuration. The springs are excited via the use of an electromagnetic coil, which applies a force to a small magnet connected to springs. Varying the signal in the coil produces corresponding vibrations in the spring. The output from the system is taken with a similar configuration. A small magnetic bead oscillates with the spring at another location (usually the opposite end of the spring), and induces current in a nearby electromagnetic coil.

There has been much recent work on modelling the behavior of the spring reverberator digitally for use in a music production environment. The first attempts used an optimisation method to fit the dispersion curve of the spring with a number of allpass filters, and then used these filters within a wave-guide structure [2]. More recent work has modelled the vibration of the spring using finite difference methods [3, 4]. Attempts have also been made to produce a parametric digital spring reverberator which more closely

^{*} This work has been supported by the [MIDE program] of Aalto university, the Helsinki Graduate School in Computer Science and Engineering (HeCSE), and the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no. [203636].

[†] This work has been supported by the Academy of Finland, project no. [122815].

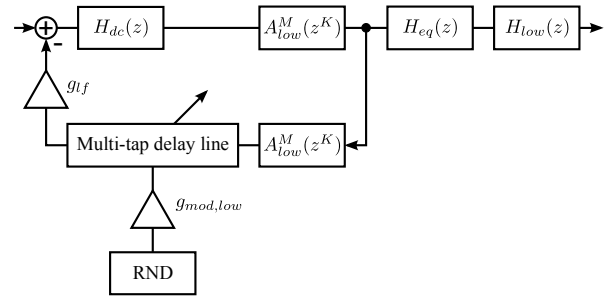


Figure 1: Low-frequency structure of parametric spring reverberation effect, based on [5].

resembles the structure of a traditional digital reverberator [5]. It is this final approach to modelling which we consider in this work.

In this work we propose a method for automatically deriving the characteristics of a spring reverb unit from a recorded spring impulse response. These characteristics are used to tune the parameters of the digital spring reverb model, much like previous work has proposed an analogous method that allows the fitting of a digital reverberator to a specific room response [6].

The paper is organised as follows. The parametric spring reverb model used for the automated calibration is briefly introduced in Section 2. Section 3 describes the proposed automated calibration methods for the parametric spring reverb model. The results of the automated calibration are discussed in Section 4. Section 5 concludes the paper.

2. THE PARAMETRIC SPRING REVERBERATION EFFECT

Basis for this work is the parametric spring reverberation effect introduced by Välimäki et al. [5]. Figure 1 shows a block diagram of the feedback structure used to produce the low-frequency chirp sequence of the spring reverberation effect (cf. Figure 2). The filter $H_{dc}(z)$ is a dc blocking filter with a cutoff frequency at 40 Hz [5]. The filter $A_{low}^M(z^K)$ is a spectral delay filter [7] consisting of M cascaded allpass filters. Each allpass filter section is an interpolated stretched allpass filter, which is composed of a Schroeder allpass filter with an embedded delay line of $K - 1$ samples and a first-order fractional-delay allpass filter to implement a delay equal to 1 plus the decimal part of K (in samples). The impulse response of the spectral delay filter imitates the first low-frequency chirp appearing in the response of a spring reverb unit.

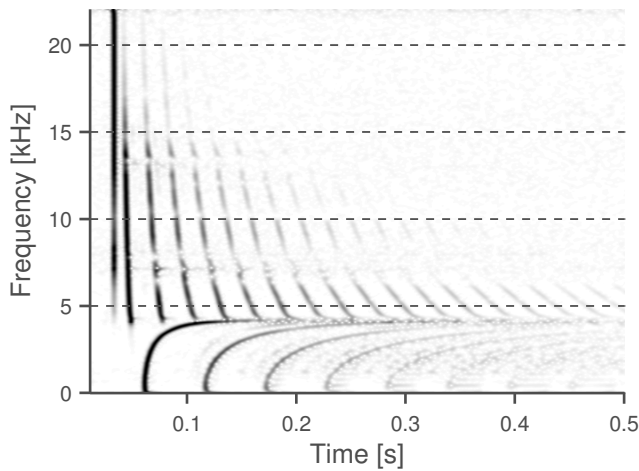


Figure 2: Spectrogram of one spring of the Leem Pro KA-1210 spring reverb. The spectrogram is normalised for each frequency bin, to improve the visibility of the chirp structure.

A multi-tap delay line models the propagation and scattering of waves in a helical spring. It contains a long delay line with several extra output taps, which introduce pre-echos in the temporal response and small variations in the loop gain of the system [5]. A random modulation is applied to the delay line to introduce blurring of the response over time [5]. Finally, the output of the model is processed with two filters, $H_{eq}(z)$ and $H_{low}(z)$, which are a second-order resonator and a low-pass filter, respectively. An allpass cascade A_{low} is inserted into the feedback path to model the dispersion of each reflection when traversing the spring backwards, as proposed by Parker et al. [8].

To produce the high-frequency chirp sequence in the impulse response of the spring reverberator, a feedback structure similar to the one depicted in Figure 1 is used in the model, as proposed in [5]. It is considered less perceptually important [5]. The all-pass cascade in the feedback path is omitted for the high-frequency feedback structure, to reduce the computational complexity of the model. The reader is referred to [5] for a more detailed description of the parametric spring reverb model.

3. AUTOMATED CALIBRATION

The starting point for the automated calibration of the parametric spring reverberation effect is the spectrogram of the impulse response to be modelled. Figure 2 shows the spectrogram of one spring of the Leem Pro KA-1210 spring reverb. It is obtained via an 8192-point short-time Fourier Transform (STFT) using a Blackman window with a hopsize of 8 samples. The spectrogram is normalised at each frequency bin, to enhance the visibility of the high-frequency chirp structure.

The calibration of the model is performed in three steps: First, the pulse delay of the low-frequency chirp structure and the transition frequency are determined. These are the perceptually most important parameters of the model [4]. In the following steps, the parameters for the low- and high-frequency chirp structures are determined separately.

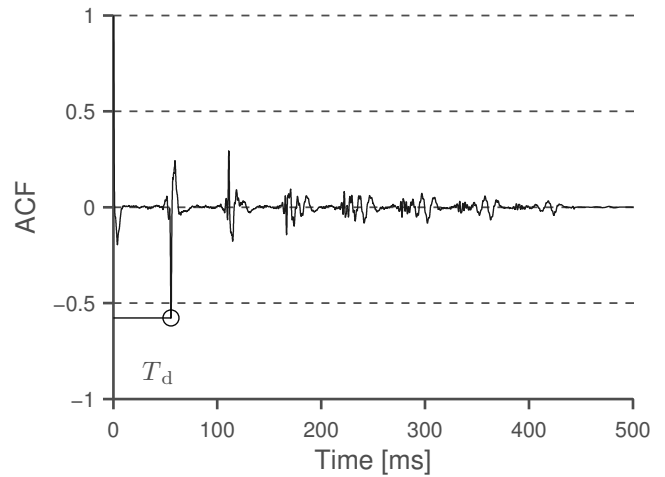


Figure 3: Autocorrelation function. The pulse delay is estimated at the maximum absolute value, $T_d = 55.6$ ms.

3.1. Determine pulse delay and transition frequency

3.1.1. Pulse delay

The time domain representation of a spring reverb impulse response is dominated by the low-frequency pulse sequence, which contains most of the energy [5]. The pulses recur at regular intervals, given by the pulse delay T_d . It can be derived from the maximum absolute value of the autocorrelation of the impulse response (see Figure 3). For the given impulse response, T_d is estimated as 55.6 ms.

3.1.2. Transition frequency

The transition frequency F_c is defined as the cutoff frequency of the low-frequency pulse series [4]. As can be seen from the spectrogram in Figure 2, the low-frequency pulses overlap in time around the cutoff frequency, and thus cannot be distinguished from one another.

To determine F_c , the normalised autocorrelation of the spectrogram is calculated for each frequency bin along the time axis. The spectrogram can be represented as a matrix \mathbf{S} , with rows corresponding to frequency bins and columns corresponding to time instants. The autocorrelation of each row in \mathbf{S} is calculated and normalised to one at lag zero. The result of this calculation is shown in Figure 4 (left). As can be seen, Figures 2 and 4 exhibit a similar pulse structure: For each frequency, the delay caused by dispersion and propagation in the spring is constant between pulses. The autocorrelation around the transition frequency contains no distinct peaks, as around this frequency the pulses overlap in time. Since the autocorrelation is normalised to one at lag zero, the mean value of the autocorrelation calculated over time at each frequency bin is a measure for the periodicity of the impulse response with respect to frequency. The mean value exhibits a peak at the transition frequency, where no distinct pulses are visible in the spectrogram, i.e., the periodicity is lowest (see Figure 4, right). Here, F_c is estimated as 4216 Hz. The method works well also for impulse responses with more than one spring, if the springs share a transition frequency.

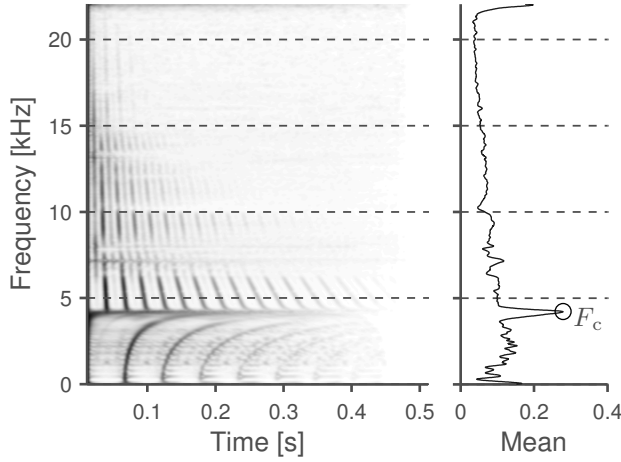


Figure 4: Normalised autocorrelation of spectrogram and mean of autocorrelation. The transition frequency is estimated at the maximum (\circ), $F_c = 4216$ Hz.

3.2. Low-frequency chirps

3.2.1. Gain factor

Figure 5 depicts the energy decay curve of the original impulse response. It is obtained via a backward integration of energy, the so-called Schroeder integration [9]. The black dots are T_d -spaced, indicating the low-frequency pulse positions, which dominate the impulse response [5]. In the parametric model used in this paper [5], the energy decay rate of the impulse response is modelled by applying a constant gain factor g_{lf} to the pulse series (cf. Figure 1). It determines the attenuation of each reflected pulse with respect to the previous. The value of g_{lf} is estimated by fitting a line through the energy decay curve.

For the perceived reverberance, the early decay time (EDT) from 0 dB to -10 dB is considered particularly important [10]. To model the EDT, a line is fitted through the energy decay curve at the pulse positions, from the first pulse to the pulse where the impulse response energy decays below -10 dB. The fitting is implemented via the polyfit function in Matlab (cf. black line in Figure 5). The decay per pulse d in dB is obtained as

$$d = mT_d, \quad (1)$$

where m is the slope of the fitted line and T_d is the pulse delay. From the pulse decay d , the gain factor g_{lf} is obtained via

$$g_{lf} = K10^{(\frac{d}{20})}, \quad (2)$$

with

$$K = \text{sgn} \{ \max[ACF(x)] \}, \quad (3)$$

i.e., the sign of the maximum of the autocorrelation function ACF of the spring impulse response x (cf. Figure 3). To compensate for additional attenuation of the pulses introduced by the modulation and linear interpolation in the delay line, the gain factor g_{lf} is multiplied by a constant of 1.2. For the given impulse response, this yields $g_{lf} = -0.64$.

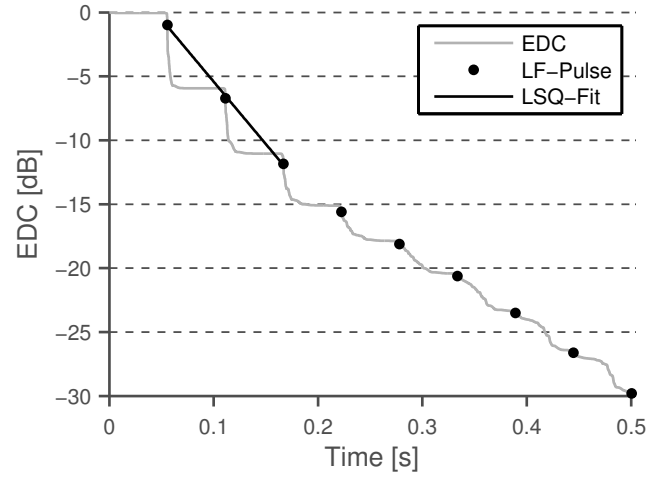


Figure 5: Energy decay curve (grey line) and low-frequency pulse positions (black dots). A least-squares fit (black line) indicates a decay rate of about -5.4 dB per pulse, during the early decay from 0 dB to -10 dB.

3.2.2. Spectral delay filter calibration

A crucial step in the calibration of the model is determining the parameters for the spectral delay filter. The goal is to tune the output of the filter to match the dispersion characteristics of the real spring. The transfer function of the stretched interpolated allpass cascade used in the model [5] to generate the chirps is given by

$$A_M(z) = M \frac{a_1 + A_{fd}(z)z^{-K_1}}{1 + a_1 A_{fd}(z)z^{-K_1}}, \quad (4)$$

where A_{fd} is a fractional delay allpass filter with

$$A_{fd}(z) = \frac{a_2 + z^{-1}}{1 + a_2 z^{-1}}, \quad (5)$$

and

$$K_1 = \text{round}(K) - 1. \quad (6)$$

M is the number of allpass sections in the cascade, a_1 is the allpass coefficient, and K the stretching factor determining the cutoff frequency of the chirps. The fractional delay filter is necessary to implement a nonintegral stretching factor K , to accurately obtain the desired transition frequency of the chirp structure. The parameter a_2 can be derived from the nonintegral part of the stretching factor (for details, see [5]). The parameters M , a_1 and K of the allpass cascade are obtained by iteratively fitting the allpass cascade to the first chirp in the spectrogram. The procedure is applicable to spring reverbs with one or more springs, therefore the general case of N springs is considered in the following.

First, peaks in the spectrogram are extracted at each frequency bin, up to the transition frequency. After low-passing the rows of the spectrogram matrix \mathbf{S} , which correspond to frequency bins (cf. Section 3.1.2), the peak locations at each frequency bin are detected as zero crossings of the derivative of each row. Next, the peaks in \mathbf{S} are grouped to connected segments, by identifying sequences of peaks that form a connected line in the spectrogram. Finally, these peak segments are grouped to chirps in an iterative process, using the output of the allpass cascade as a model for the chirps:

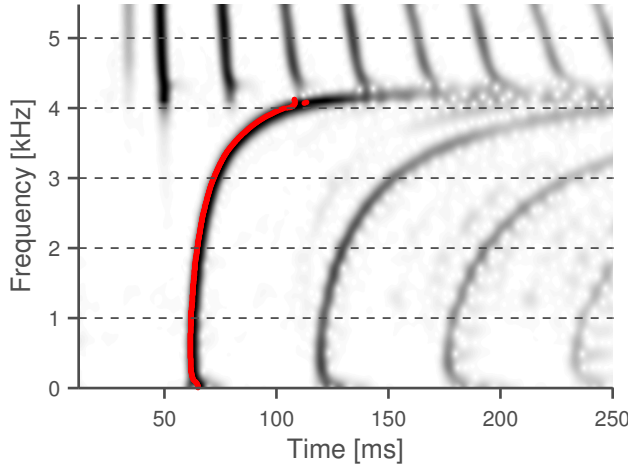


Figure 6: The first chirp (red) is identified through iterative fitting of an allpass cascade to peak segments of the spectrogram.

1. Fit the output of an allpass cascade to the longest peak segment in the spectrogram, using nonlinear least squares fitting via the `lsqnonlin` function in Matlab. This yields a rough estimate \hat{c}_i of the i -th chirp in the spectrogram.
2. Find peak segments that lie close to \hat{c}_i , update \hat{c}_i to include those segments and re-fit the allpass cascade to the new \hat{c}_i .
3. Repeat step 2 while peak segments close to \hat{c}_i are found.
4. Remove all peak segments allocated to any chirp estimate \hat{c}_i , and repeat steps 1–3, until all chirps $\hat{c}_{1...N}$ are extracted.

This process identifies the chirps in the spectrogram produced by the N springs and directly yields the parameters of the allpass cascades to model their estimates $\hat{c}_{1...N}$. The pulse delay $T_{d,i}$ of each chirp is given as

$$T_{d,i} = T_d + 2\Delta T_{0,i}, \quad (7)$$

where T_d is the pulse delay (cf. Section 3.1.1), and $T_{0,i}$ is the offset of the i -th chirp from the first chirp visible in the spectrogram. The result of the chirp extraction for the given impulse response is shown in Figure 6.

3.2.3. Chirp equalisation

The spectrum of the first chirp is obtained by calculating the Fourier Transform of the impulse response from 0 to $2T_d$. To approximate the spectral shape of the chirp, a second-order IIR filter is used. Its transfer function is stretched by replacing the unit delays of the filter structure with a delay line of length K [5]. The filter parameters consist of the stretching factor K , the frequency F_{peak} of the peak in the transfer function, and the -3 dB bandwidth B of the peak. Nonlinear least-squares fitting is used to fit the frequency response of the equalisation filter to the spectrum of the first chirp (see Figure 7).

3.3. High-frequency chirps

The high-frequency chirp sequence is considered less perceptually important [5]. Therefore, a simpler approach towards calibration can be taken than for the low-frequency structure. Based on

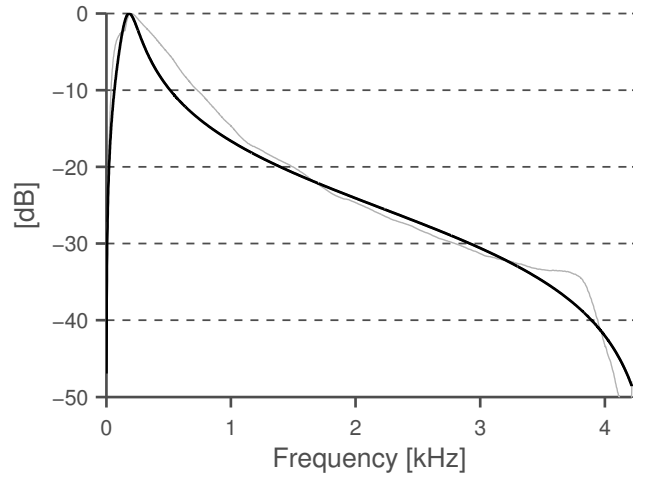


Figure 7: Spectrum of first pulse (light grey) and fitted equalisation filter: $F_{\text{peak}} = 183\text{Hz}$, $B = 146\text{Hz}$, $K = 5$.

the assumption of a regular pulse structure, the chirps can be extracted from the autocorrelation of the spectrogram matrix \mathbf{S} (cf. Section 3.1.2). This procedure emphasises periodic energy components in the spectrogram, whilst suppressing nonperiodic components, such as the first pulse appearing in the spectrogram above 5 kHz (see Figure 2).

To detect the first chirp of the high-frequency pulse series in the autocorrelation of the spectrogram, the locations of the two largest peaks are determined for each frequency bin, using the method described in Section 3.2.2. To eliminate all peaks not belonging to the periodic pulse series, a simple check is performed: All peaks in the autocorrelation belonging to a periodic chirp in the pulse series must have a corresponding peak at twice the delay (cf. Figure 8). The peaks in the autocorrelation passing this check indicate the dispersion and delay induced to a periodic pulse after traversing the spring twice: Dividing the locations of the peaks by two yields the form of the first chirp in the spectrogram, at half the delay between adjacent chirps. It is modelled by fitting an allpass cascade via nonlinear least-squares fitting.

For simplicity, the decay rate of the high-frequency chirps is obtained by multiplying the decay rate of the low-frequency chirp sequence with a constant factor. Based on inspection of the spectrograms of spring reverb impulse responses, we chose 1.3 for the factor, assuming that the decay rate per pulse is about 30% lower for the high-frequency than for the low-frequency chirp sequence. This yields a high-frequency gain factor $g_{\text{hf}} = -0.83$.

4. DISCUSSION

Figure 9 allows a visual comparison of the spectrogram of a real spring reverb unit and the spectrogram of the digital parametric model after automated calibration.

The main perceptual parameters, i.e., the transition frequency and pulse delay of the low-frequency chirp sequence are modelled quite accurately. The form of the low-frequency chirps is captured well, although the group delay close to the transition frequency appears to be larger in the real impulse response than in the model. As a result of the linear approximation of the early decay time (EDT, cf. Figure 5), the energy of the low-frequency chirps decays

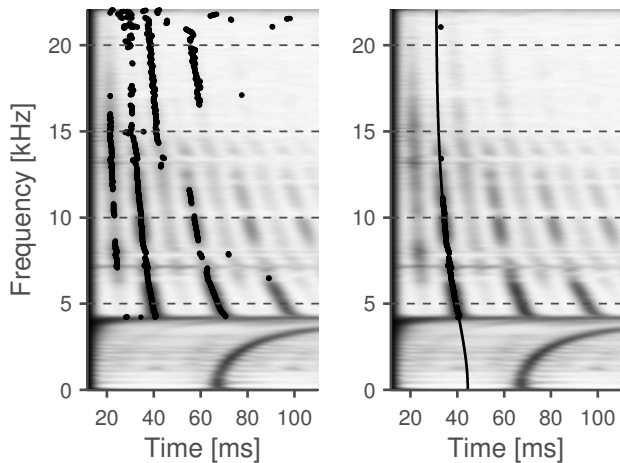


Figure 8: Left: First and second peaks in normalised autocorrelation of spectrogram. Right: Peaks of first pulse after outlier removal (\circ) and fitted allpass cascade (—): $a_1 = -0.34$, $M = 189$.

slower in the real response than in the model. An auditory comparison reveals a marked difference in timbre between the real and modelled low-frequency chirp sequence. This may partly be due to the coarse approximation of the chirp spectrum (cf. Figure 7), which results in a discrepancy between the desired and modelled spectrum, particularly below 1.5 kHz and above 3.5 kHz. Using a higher-order equalisation filter in the digital model might help tackle this problem. Furthermore, the acoustic quality of the increasing diffuseness of successive chirps could be modelled more accurately, for example by replacing the delay line modulation with an automatically calibrated digital reverberator [5].

The main characteristics of the high-frequency chirp structure are successfully reproduced by the model. The form of the first high-frequency chirp is modelled accurately. The following chirps are more strongly dispersed in the real impulse response than in the model. To model the dispersion characteristics more accurately, an allpass cascade could be inserted in the feedback path of the high-frequency structure of the model. It is omitted here since it is not considered perceptually important and reduces the computational load of the model considerably [5]. The decay rate of the chirps seems to be slightly lower in the real impulse response than in the model. This is a result of the linear approximation of the decay rate in the parametric model.

The automated calibration was performed without constraints in terms of the computational complexity of the digital parametric model. The computational load is dominated by the allpass chains. To lower the computational load of the model, an upper limit can be set to the length of the allpass chains fitted by the optimisation algorithms described in Sections 3.2.2 and 3.3. As an example, we set the maximum length of the low-frequency allpass cascade to $\max\{M_{low}\} = 100$. Table 1 presents an overview of the calibrated model parameters for the Leem Pro KA-1210 spring reverb. The first column contains parameter values obtained through manual and semi-automated calibration [5]. The middle column contains the values obtained using the automated calibration proposed in this paper, without constraints in terms of computational complexity. The last column presents the values obtained with an upper limit on the length of the low-frequency allpass chain. There

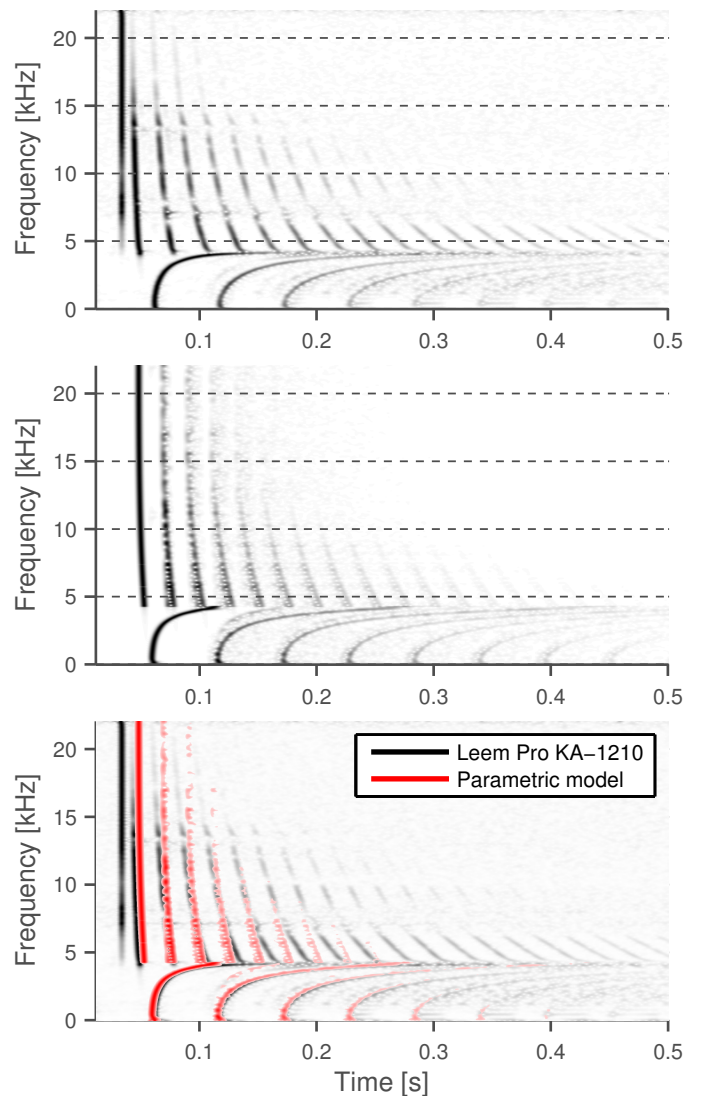


Figure 9: Spectrogram of the impulse response of the Leem Pro KA-1210 (top) and of the parametric model with automated calibration (middle). The bottom graph shows the model spectrogram as an overlay (red) onto the spectrogram of the Leem Pro KA-1210 (greyscale).

is good correspondence between the manual and automated calibration of the values T_d and F_c . The form of the modelled chirps is determined by the parameters a and M , as well as $F_{c,lf}$ in case of the low-frequency chirp structure. If an upper limit is set to M , the values of a and $F_{c,lf}$ change such as to obtain an optimal fit of the modelled chirp. The spectrogram of the model with limited allpass cascade length is shown in Figure 10. A visual comparison reveals no substantial differences to the spectrogram of the model with unconstrained cascade length, indicating that the restriction is a viable option to reduce computational load without major perceptual impact. However, the form of the first chirp is modelled less accurately with the reduced allpass chain length. Figures 11 and 12 demonstrate the usage of the automated calibration for spring

	Leem Pro KA-1210, Spring 1		
	Manual [5]	Automated	Automated*
T_d	0.056	0.056	0.056
F_c	4300	4216	4216
$F_{c,lf}$	4300	4980	4275
M_{low}	100	318	100
a_1	0.62	0.69	0.63
g_{lf}	-0.8	-0.64	-0.64
M_{high}	200	189	189
a_{high}	-0.6	-0.34	-0.34
g_{hf}	-0.77	-0.83	-0.83

* With $\max \{M_{low}\} = 100$.

Table 1: Comparison between manual and automated calibration of the parametric spring reverberation effect. The transition frequency F_c refers to the transition frequency of the impulse response, whereas $F_{c,lf}$ refers to the transition frequency of the fitted allpass cascade. The optimised calibration results were obtained by setting an upper limit to M_{low} .

reverbs with two and three springs. In both cases the main features of the spectrogram of the real impulse response are captured successfully by the proposed automated calibration, and reproduced by the parametric model. However, the modelling accuracy is inferior to the single-spring example, since the parameters of each spring are extracted from a single response of the whole unit.

5. CONCLUSION

In this paper, an approach was proposed to automatically calibrate parameters of a spring reverberation model. A slight modification of a previously presented spring reverberation model is used [5], which produces impulse responses containing the same basic features appearing in responses measured from spring reverb units.

Signal processing methods to estimate the values of several parameters of the spring reverb model were suggested. All estimations are based on a measured impulse response of a spring reverberation unit or its spectrogram. First, the time delay between repetitive pulses appearing at low frequencies was estimated by detecting the first peak in the absolute value of the autocorrelation function of the measured response. Next, the transition frequency, which corresponds to the cutoff point of the low-frequency chirp sequence, was estimated from the autocorrelation function of the spectrogram. The maximum of the mean of the autocorrelation function appeared to indicate the transition frequency. The value of the feedback gain factor was computed from the early decay time, which we estimated as the difference of time instants, where the energy decay curve passes the -10 dB level.

A spectral delay filter consisting of a chain of first-order allpass filters imitates the shape of the first chirp in the impulse response. The number of cascaded allpass filters and their filter coefficient value, which is the same for all filters in the cascade, were chosen by fitting the output of the allpass cascade via an iterative procedure employing the nonlinear least squares method. The spectral delay filter was equalised by fitting the parameters of a resonant second-order filter to the magnitude spectrum of the chirp, using the nonlinear least-squares method.

Model parameters for the high-frequency chirp sequence are

also extracted from the autocorrelation of the spectrogram, although the data is noisier than that containing the low-frequency chirps. No equalisation is performed for the high-frequency chirps.

The proposed calibration methods can be applied to a response of a single-spring unit or to one with several parallel springs. The calibrated parameter values were compared against the manually and semi-automatically calibrated values presented in [5]. It was observed that slightly different but similar values are obtained. The optimal number of allpass sections in the spectral delay filter for the low-frequency chirp can become very large, such as about 300, leading to a high computational load in the implementation. For this reason a constrained optimisation was tested in which the maximum number of filter sections is limited to 100. This can lead to a sufficiently good fit and to a reasonable number of filtering operations per sample.

Recently, Parker has proposed multirate and subband techniques to reduce the computational cost of the parametric spring reverberation model [11]. These ways to improve the computational efficiency are suggested to be applied after parameter values have been calibrated using methods proposed in this paper.

6. REFERENCES

- [1] L. Hammond, "Electrical musical instrument," US Patent No. 2230836, 1941.
- [2] J. S. Abel, D. P. Berners, S. Costello, and J. O. Smith, "Spring reverb emulation using dispersive allpass filters in a waveguide structure," in *Audio Engineering Society Convention 121, San Francisco*, 2006.
- [3] S. Bilbao and J. Parker, "A virtual model of spring reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 799–808, 2010.
- [4] J. Parker and S. Bilbao, "Spring reverberation: A physical perspective," in *Proceedings of the International Digital Audio Effects Conference, Como, Italy*, 2009.
- [5] V. Välimäki, J. Parker, and J. S. Abel, "Parametric spring reverberation effect," *Journal of the Audio Engineering Society*, vol. 58, no. 7/8, pp. 547–562, 2010.
- [6] J.-M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Audio Engineering Society Convention 90, Paris*, 1991.
- [7] V. Välimäki, J. S. Abel, and J. O. Smith, "Spectral delay filters," *Journal of the Audio Engineering Society*, vol. 57, no. 7/8, pp. 521–531, 2009.
- [8] J. Parker, H. Penttinen, S. Bilbao, and J. S. Abel, "Modeling methods for the highly dispersive slinky spring: A novel musical toy," in *Proceedings of the International Digital Audio Effects Conference, Graz, Austria*, 2010.
- [9] M. R. Schroeder, "New method of measuring reverberation time," *The Journal of the Acoustical Society of America*, vol. 37, no. 3, pp. 409–412, 1965.
- [10] J.S. Bradley, "Review of objective room acoustics measures and future needs," *Applied Acoustics*, vol. 72, no. 10, pp. 713–720, 2011.
- [11] J. Parker, "Efficient dispersion generation structures for spring reverb emulation," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, 2011, Article ID 646134, 8 pages.

7. APPENDIX

This appendix presents the spectrograms of the real impulse response and of the parametric model with automated calibration for spring 1 of the Leem Pro KA-1210 spring reverb (see Figure 10), a no-name spring reverb containing two springs (see Figure 11), and a Mesa Boogie spring reverb containing three springs (see Figure 12). Sound samples are available for download at <http://www.tml.tkk.fi/~hannes/DAFx2011/>.

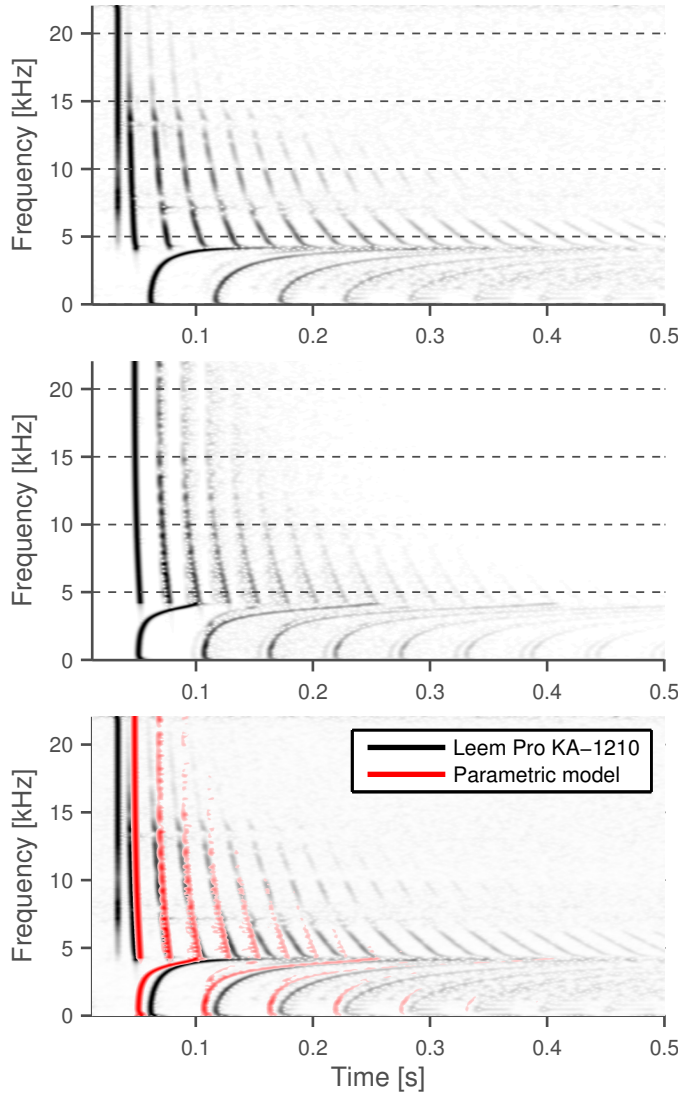


Figure 10: Spectrogram of impulse response of spring 1 of Leem Pro KA-1210 (top) and of the parametric model with automated calibration (middle), with length of the low-frequency allpass cascade limited to $\max \{M_{\text{low}}\} = 100$. The bottom graph shows the model spectrogram as an overlay (red) onto the spectrogram of the real unit (greyscale). Although the main perceptual aspects of the low-frequency chirp sequence are captured, the limited length of the allpass chain deteriorates the modelling accuracy of the form of the low-frequency chirps. This is a trade-off for reducing the computational complexity by limiting the length of the allpass cascade.

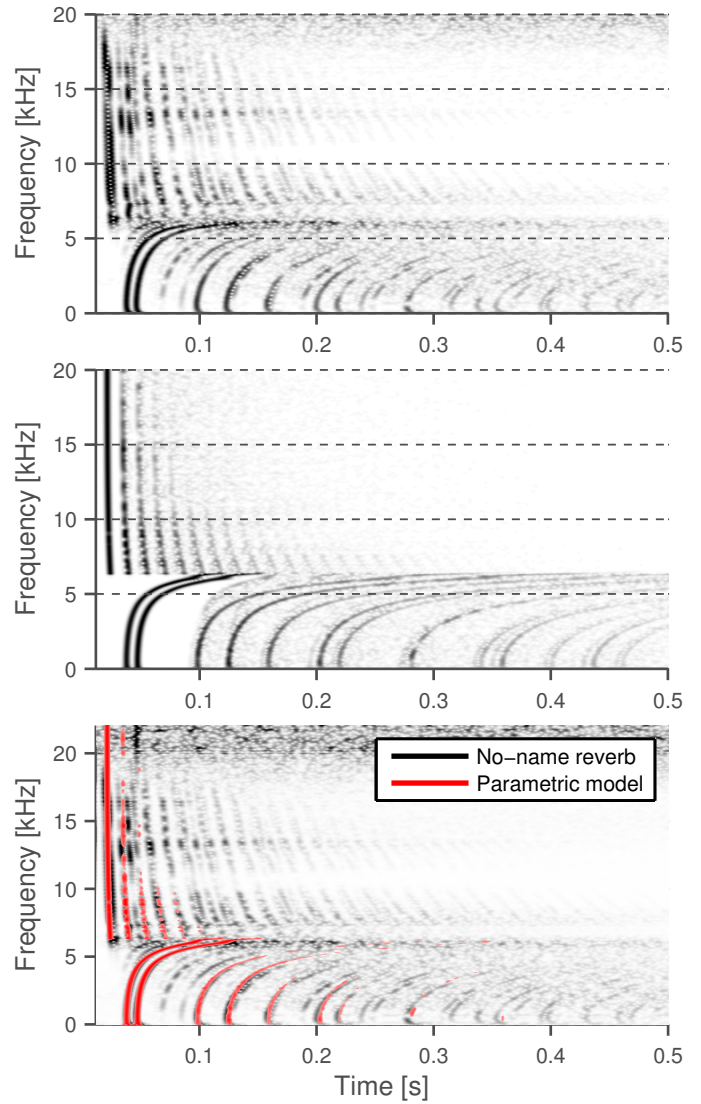


Figure 11: Spectrogram of impulse response of a no-name spring reverb with two springs (top) and of the parametric model with automated calibration (middle). The bottom graph shows the model spectrogram as an overlay (red) onto the spectrogram of the real spring reverb unit (greyscale). The main features of the real impulse response are captured well. However, additional low-frequency chirp reflections and some details of the high-frequency chirp sequence are not reproduced by the model. The modelling accuracy could presumably be improved if individually measured responses of both springs were used for the automated calibration. The same holds for the model response shown in Figure 12. It is more difficult to derive parameters of all springs from a single response measured of the whole unit than using individually measured responses for each spring.

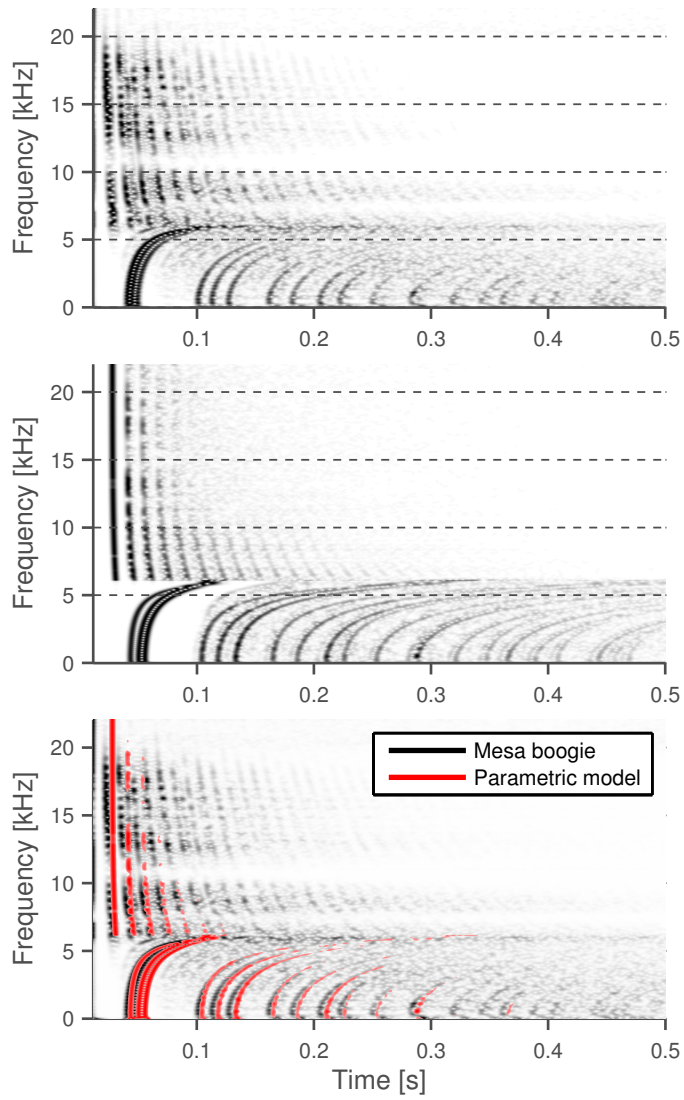


Figure 12: Spectrogram of impulse response of Mesa Boogie spring reverb with three springs (top) and of the parametric model with automated calibration (bottom). The bottom graph shows the model spectrogram as an overlay (red) onto the spectrogram of the real spring reverb (greyscale). All three low-frequency chirp sequences are modelled relatively accurately, although there are discrepancies regarding the form of the first modelled chirps and the details of the high-frequency chirp sequence.

LYAPUNOV STABILITY ANALYSIS OF THE MOOG LADDER FILTER AND DISSIPATIVITY ASPECTS IN NUMERICAL SOLUTIONS

Thomas Hélie,

Equipe Analyse/Synthèse, IRCAM - CNRS UMR 9912 - UPMC

1, place Igor Stravinsky,

75004 Paris, France

thomas.helie@ircam.fr

ABSTRACT

This paper investigates the passivity of the Moog Ladder Filter and its simulation. First, the linearized system is analyzed. Results based on the energy stored in the capacitors lead to a stability domain which is available for time-varying control parameters meanwhile it is sub-optimal for time-invariant ones. A second storage function is proposed, from which the largest stability domain is recovered for a time-invariant Q-parameter. Sufficient conditions for stability are given. Second, the study is adapted to the non-linear case by introducing a third storage function. Then, a simulation based on the standard bilinear transform is derived and the dissipativity of this numerical version is examined. Simulations show that passivity is not unconditionally guaranteed, but mostly fulfilled, and that typical behaviours of the Moog filter, including self-oscillations, are properly reproduced.

1. INTRODUCTION

The Moog Ladder Filter [1] is an analog audio device which is appreciated by many musicians because of its intuitive control, its sound singularity and its typical self-oscillating behaviour at high feedback-loop gains. This nonlinear analog circuit has been deeply studied and simulated using distinct methods [2, 3, 4, 5, 6, 7, 8].

Approaches based on energy and passivity considerations have proved to be relevant for simulating nonlinear systems, including applications to sound synthesis. This issue is of main importance for conservative systems whose simulation must neither diverge nor introduce parasitic dampings. These considerations have yet motivated works on e.g. nonlinear strings, plates and shells (see e.g. [9, 10, 11]). This approach is also worthwhile for dissipative systems, especially when they are able to reach conservative and self-oscillating behaviours. Thus, the filter of the EMS VCS3 synthesizer has been simulated using a decomposition of the circuit into modules which preserves passivity properties [12]. Moreover, these methods usually allow to derive simulations which are compatible with real-time computations usable by musicians.

In this paper, a study on the Moog filter passivity is performed from which a stability criterion is deduced according to the so-called Lyapunov stability analysis [13]. Lyapunov functions based on (a) the natural energy stored in the capacitors, (b) energies conveyed by eigenvectors of the linearized system and (c) some modified versions adapted to the nonlinear dynamics are considered. They allow to characterize stability domains as well as dissipated quantities. These features are applied to the discrete-time dynamics analysis of the bilinearly-transformed version of the system.

The outline of this paper is as follows. Section 2 recalls the

circuit equations and provides a state-space representation of the system. Section 3 refreshes the definition of passivity, of Lyapunov functions as well as basic results on stability analysis. The stability analysis is performed, first, on the linearized version of the circuit in section 4, and then, on the original nonlinear circuit in section 5. Finally, in section 6, a dissipativity indicator especially designed for the bilinear transform is deduced, which allows to characterize the passivity preservation in simulated results.

2. CIRCUIT AND EQUATIONS

2.1. Circuit description

The Moog ladder filter is a circuit composed of (see Fig. 1 a-d): a driver, a cascade of four filters involving capacitors C , differential pairs of NPN-transistors and an additional feedback loop (detailed in (e)). Following [1], transistors are LM3046 or BC109a,b,c, po-

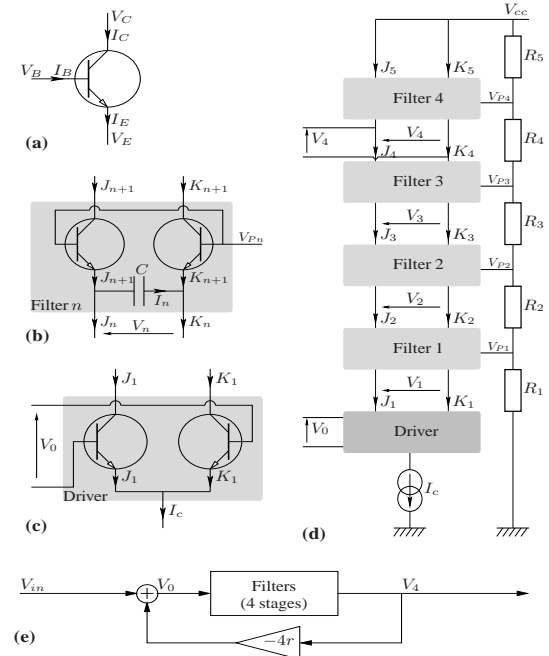


Figure 1: Circuits : (a) NPN transistor, (b) single-stage filter, (c) driver, (d) four-stages Moog ladder filter without loop and (e) Complete filter including a feedback-loop gain $-4r$.

larization voltages are $V_{P1} = 2.5$ V, $V_{P2} = 4.4$ V, $V_{P3} = 6.3$ V,

$V_{P4} = 8.2$ V, $V_{cc} = 15$ V corresponding to $R_1 = 350$ Ohms, $R_2 = R_3 = R_4 = 150$ Ohms, $R_5 = 390$ Ohms. Capacitances are $C = 27$ nF for the Moog Prodigy and $C = 68$ nF for the MiniMoog synthesizer. Moreover, I_c is a voltage-controlled current which tunes a cut-off frequency and parameter r (in Fig. 1e) is a the voltage-controlled gain which monitors the resonance (no resonance if $r = 0$, resonance with infinite Q -factor (self-oscillations) if $r = 1$ [2]).

2.2. Circuit equations, energy and power balance

Denote q_n , I_n and V_n the charge, the current and the voltage of the capacitor in the n -th stage, respectively. The circuit equations are (see e.g. [8, § II] for a detailed derivation for transistor pairs)

$$\text{Capacitor law: } q_k = C V_k, \quad (1)$$

$$\text{Capacitor current: } \frac{dq_k}{dt} = I_k, \quad (2)$$

$$\text{Transistor pair: } I_k = -\frac{I_c}{2} \tanh \frac{V_k}{2V_T} + \frac{I_c}{2} \tanh \frac{V_{k-1}}{2V_T}, \quad (3)$$

$$\text{Loop: } V_0 = V_{in} - 4rV_4. \quad (4)$$

where the thermal voltage is $V_T = k_b T / q \approx 25.85$ mV at temperature $T = 300$ K ($k_b = 1.38 \cdot 10^{-23}$ J/K is the Boltzmann constant, $q = 1.6 \cdot 10^{-19}$ C is the electron charge).

The total energy which is stored in the capacitors is given by

$$E = \sum_{k=1}^4 \frac{q_k^2}{2C}, \quad (5)$$

the time derivative of which yields the following power balance

$$\frac{dE}{dt} = \sum_{k=1}^4 \frac{q_k}{C} \frac{dq_k}{dt} = \sum_{k=1}^4 V_k I_k, \quad (6)$$

which can be formulated as a function of $V_{0,\dots,4}$ using (3).

2.3. Dimensionless version

Consider the dimensionless quantities

$$x_k = \frac{V_k}{V^*} = \frac{q_k}{q^*}, \quad i_k = \frac{I_k}{I^*}, \quad u = \frac{V_{in}}{V^*}, \quad e = \frac{1}{2} \frac{E}{E^*},$$

with $V^* = 2V_T$, $q^* = C V^*$, $I^* = \frac{I_c}{2}$ and $E^* = \frac{(q^*)^2}{2C}$. Then, (1) becomes trivial ($x_k = x_k$) since x_k characterizes both the voltage and the charge of capacitors. Moreover, (2-6) become, respectively, $\frac{1}{\omega} \frac{dx_k}{dt} = i_k$, $i_k = -\tanh x_k + \tanh x_{k-1}$, $x_0 = x_4 - 4ru$,

$$e = \sum_{k=1}^4 \frac{x_k^2}{2}, \quad \text{and} \quad \frac{de}{dt} = \sum_{k=1}^4 x_k \frac{dx_k}{dt} = \omega \sum_{k=1}^4 x_k i_k, \quad (7)$$

which can be formulated as a function of $x_{1,\dots,4}$ and u and where $\omega = I^*/q^* = I_c/(4C V_T)$ (in s^{-1}).

2.4. State-space representation and parameters

The dimensionless versions of (2-4) for $k = 1, \dots, 4$ yield the state-space representation with state x and input u , given by

$$\frac{1}{\omega} \frac{dx}{dt} = f(x, u) \text{ with } f(x, u) = \begin{bmatrix} -\tanh x_1 + \tanh(u - 4r x_4) \\ -\tanh x_2 + \tanh x_1 \\ -\tanh x_3 + \tanh x_2 \\ -\tanh x_4 + \tanh x_3 \end{bmatrix}, \quad (8)$$

where the cutoff angular frequency $\omega = \frac{I_c}{4C V_T}$ has been extracted from f for sake of simplicity in the following derivations.

Remark 1. In all the following equations (unless otherwise mentioned), the control parameters ω and r can depend on time and lie in $\omega \in \mathbb{R}_+^*$, $r \in [0, 1]$ (the case $\omega = 0$ which yields no dynamics $\frac{dx}{dt} = 0$ so that $x(t) = x(0)$ is discarded here). Note also that the signal which is usually used as the output of the filter corresponds to x_4 .

3. RECALLS ON PASSIVITY, STABILITY AND LYAPUNOV ANALYSIS

This section recalls some basic results about passivity, stability and Lyapunov analysis. The detailed theory can be found in [13, chapter 6].

Definition. Consider a system with state-space representation $\frac{dx}{dt} = f(x, u)$, output $y = \phi(x, u)$, where $f : \mathbb{R}^k \times \mathbb{R}^p \rightarrow \mathbb{R}^k$ is sufficiently regular (locally Lipschitz), $\phi : \mathbb{R}^k \times \mathbb{R}^p \rightarrow \mathbb{R}^p$ is continuous, $f(0, 0) = 0$ and $\phi(0, 0) = 0$. This system is said to be passive if there exists a continuously differentiable positive semidefinite function $\mathcal{V}(x)$ called the storage function such that $\frac{d}{dt}(\mathcal{V}(x)) \leq y^T u$. It is said to be *strictly passive* if

$$\frac{d}{dt}(\mathcal{V}(x)) \leq -\psi(x) + y^T u, \quad (9)$$

for some positive definite function ψ .

Such a storage function \mathcal{V} is called a *Lyapunov function*. It can be assimilated to an energy of the system and ψ to a dissipated power. One interest of the following approach is that it can be used for linear, nonlinear and possibly time-varying systems.

Links between passivity and stability. The passivity is a practical tool to examine some system stability aspects. It leans on the two following key points:

- when the excitation of the system stops ($u = 0$), the positive storage function \mathcal{V} stops increasing (passive system) or even decreases as long as $x \neq 0$ (strictly passive system).
- If $\mathcal{V}(x(t))$ is bounded, then x lives inside a closed bounded set: it cannot diverge. Moreover, since \mathcal{V} is continuous and definite, if it decreases towards 0, then x also tends towards 0 (global asymptotic stability).

This very last case necessarily occurs for strictly passive systems with $u = 0$ if \mathcal{V} is radically bounded [13].

In short, this can be summarized as follows: *if a system with a suitable energy (\mathcal{V}) continuously dissipates some positive power, the system dynamics is bounded. Moreover, if function \mathcal{V} is radically bounded, the dynamics eventually tends to a steady state ($x_0 = 0$ for $u = 0$) which is stable.*

Passivity of systems in practice. Given a storage function \mathcal{V} , the passivity can be examined by deriving

$$\mathcal{P}(x, u) \stackrel{\text{def.}}{=} \left(\partial_{(x^T)} \mathcal{V}(x) \right) f(x, u) \left(= \frac{d}{dt} \mathcal{V}(x) \right) \quad (10)$$

which represents a *power* ($\partial_{(x^T)}$ denotes the partial derivatives w.r.t. the row vector x^T). Indeed, if $\mathcal{P}(x, u)$ can be written as

$$\mathcal{P}(x, u) = -\psi(x) + \phi(x, u)^T u \quad (11)$$

the passivity is obtained (with equality in (9)) w.r.t to the output $y = \phi(x, u)$.

4. LINEARIZED SYSTEM: STABILITY, ENERGY AND LYAPUNOV ANALYSIS

4.1. System equations, transfer matrix and pole analysis

4.1.1. Linear state-space equation

For $u = 0$, the unique equilibrium point of system (8) is zero. The linearized system around $(x, u) = (0, 0) \in \mathbb{R}^4 \times \mathbb{R}$ is given by

$$\frac{1}{\omega} \frac{dx}{dt} = Ax + Bu \quad \text{with } A = \partial_x f(0, 0) \text{ and } B = \partial_u f(0, 0), \quad (12)$$

$$\text{that is, } A = \begin{bmatrix} -1 & 0 & 0 & -4r \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad \text{and } B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

4.1.2. Transfer matrix and pole analysis

For time-invariant parameters ω and r , the input-to-state transfer matrix of (12) is

$$H(s) = F \left(\frac{s}{\omega} \right) \quad \text{where } F(\sigma) = (\sigma I_4 - A)^{-1} B$$

where I_4 denotes the 4×4 identity matrix, s denotes the Laplace variable and $\sigma = s/\omega$ is its dimensionless version.

The poles in the σ -complex plane are the roots of the characteristic polynomial

$$P_A(\sigma) = \det(\sigma I_4 - A) = \sigma^4 + 4\sigma^3 + 6\sigma^2 + 4\sigma + 1 + 4r.$$

For positive gains r , they are given by

$$\sigma_1 = -1 + \sqrt[4]{r} + i\sqrt[4]{r}, \quad \overline{\sigma}_1, \quad \sigma_2 = -1 - \sqrt[4]{r} + i\sqrt[4]{r} \quad \text{and} \quad \overline{\sigma}_2. \quad (13)$$

Their real parts are all strictly negative for gains $r < 1$. This condition characterizes the strict stability domain.

Remark 2. As $s = \omega\sigma$ and σ -poles do not depend on ω , changing ω modifies the cutoff frequency of the filter. But, it does not modify the quality factor of the resonance which is exclusively tuned by r . This is an appreciated particularity of the Moog filter since it makes its control easier (see [2]).

4.1.3. Conclusion on stability

For time-invariant parameters, the pole analysis reveals that the linearized system is strictly stable (poles have all a strictly negative real part) if the positive gain r satisfies $r < 1$. The case $r = 1$ corresponds to the limit of stability. The linear filter becomes unstable for $r > 1$. The stability domain does not depend on ω . Finally, for constant parameters, the strict stability maximal domain is

$$(\omega, r) \in \mathcal{D}_{max} = \mathbb{R}_+^* \times [0, 1[. \quad (14)$$

4.2. Passivity analysis based on the natural circuit energy

This section tries to restore the result (14) with a Lyapunov approach, from the passivity analysis of the linearized system. Parameters can be time-varying.

4.2.1. Passivity analysis

The energy of a physical system is a natural candidate Lyapunov function. For the (dimensionless) Moog filter, it corresponds to the sum of the energies stored in the four capacitors, given by (7), which can be rewritten as $e = \mathcal{V}(x)$ with

$$\mathcal{V}(x) = \sum_{k=1}^4 V(x_k), \quad (15)$$

where V is the energy of one capacitor

$$V(x) = \frac{1}{2} x^2, \quad (16)$$

(see § 2.3 for the conversion into dimensional physical quantities).

Using the matrix formulation $\mathcal{V}(x) = \frac{1}{2} x^T x$, (10) leads to $\mathcal{P}(x, u) = \omega (\partial_x \mathcal{V}(x))^T (Ax + Bu) = \omega (-\tilde{\psi}(x) + \tilde{\phi}(x, u)^T u)$ where

$$\tilde{\psi}(x) = -x^T A x \quad \text{and} \quad \tilde{\phi}(x, u) = Bx = \omega x_1. \quad (17)$$

The strict passivity is obtained w.r.t. the output $y = x_1$, if $\tilde{\psi}$ is positive definite. This is the case if and only if the symmetric matrix $Q = Q^t = -\frac{1}{2}(A + A^T)$ from which $\tilde{\psi}(x) = x^T Q x$ can also be defined is positive definite. This condition is equivalent to

$$\forall k \in [1, 4]_{\mathbb{N}}, \quad \det \begin{bmatrix} Q_{1,1} & \cdots & Q_{k,1} \\ \vdots & & \vdots \\ Q_{k,1} & \cdots & Q_{k,k} \end{bmatrix} \stackrel{def}{=} d_k(Q) > 0. \quad (18)$$

The matrix Q is given by

$$Q = Q^t = -\frac{1}{2}(A + A^T) = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & 2r \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 2r & 0 & -\frac{1}{2} & 1 \end{bmatrix}.$$

The sub-determinants $d_k(Q)$ are given by

$$d_1(Q) = 1, \quad d_2 = \frac{3}{4}, \quad d_3 = \frac{1}{2} \quad \text{and} \quad d_4 = \frac{5}{16} + \frac{1}{2}r - 3r^2.$$

They are all strictly positive iff $d_4 > 0$, that is, $-\frac{1}{4} < r < \frac{5}{12}$.

4.2.2. Conclusion

The Lyapunov analysis based on the natural energy does not restore the maximal stability domain \mathcal{D}_{max} given in (14) but only the subset $\mathcal{D}_{natural}$ described by

$$(\omega, r) \in \mathcal{D}_{natural} = \mathbb{R}_+^* \times [0, \frac{5}{12}[\subset \mathcal{D}_{max}. \quad (19)$$

This result which can appear contradictory at first sight is actually a quite well-known feature of the Lyapunov stability analysis, as stated below.

Remark 3. The Lyapunov stability analysis gives a sufficient condition for stability. But, it does not allow to conclude that a system is unstable if the condition is not fulfilled. Moreover, it does not guide the user in choosing the Lyapunov function (whether optimal or not). These difficulties have motivated many works to derive a candidate Lyapunov function for a given system, witnessed by a large recent bibliography on this topic. This is precisely why using the natural energy of physical systems as a Lyapunov function is usually appreciated. Unfortunately, this function is not optimal here.

At this step, it is useless to study the stability of the nonlinear system and its self-oscillating limit which is precisely known to be reached at $r = 1$: a Lyapunov function allowing to restore the stability domain \mathcal{D}_{max} must be investigated first.

4.3. Recovery of the maximal stability domain

As there is no general constructive method to design adapted Lyapunov functions, an inductive (but constructive) method is introduced below, which consists in writing the power balance in a basis of eigenvectors so that the dissipativity is straightforwardly related to the real part of eigenvalues, that is, the poles of the transfer function.

First, the method which yields a parametrized family of candidate Lyapunov functions to test is described. Second, the method is applied to the Moog system: as for the pole analysis, the obtained result restores the upper bound for time-invariant gains r .

4.3.1. Description of the proposed method

In this section, parameter r (but not ω) is assumed to be time-invariant. The method is described by the 4 following steps:

Step 1. Write the diagonal version of the system in a basis of eigenvectors $v_1, \overline{v_1}, v_2, \overline{v_2}$ associated with eigenvalues $\Sigma = (\sigma_1, \overline{\sigma_1}, \sigma_2, \overline{\sigma_2})$ defined in (13), that is,

$$\frac{1}{\omega} \frac{dx_\star}{dt} = A_\star x_\star + B_\star u$$

with $x_\star = P^{-1}x$, $A_\star = P^{-1}AP = \text{diag}(\Sigma)$, $B_\star = P^{-1}B$ and

$$P = [v_1, \overline{v_1}, v_2, \overline{v_2}]. \quad (20)$$

Step 2. Build a power balance from this state equation as follows

$$x_\star^H W_\star \left(\frac{1}{\omega} \frac{dx_\star}{dt} \right) = x_\star^H (W_\star A_\star) x_\star + x_\star^H (W_\star B_\star) u,$$

where $W_\star = \text{diag}(w_\star)$ with $w_\star \in (\mathbb{R}_+^*)^4$ can be any positive definite “diagonal weight matrix” ($x_\star^H = \overline{x_\star}^T$ denotes the hermitian, i.e. the transposed conjugate version of x_\star).

Step 3. The average of the latter equation and its hermitian version yields (recalling that u is real valued)

$$\frac{1}{\omega} \frac{de_\star}{dt} = -\tilde{\psi}_\star(x_\star) + \tilde{\phi}_\star(x_\star)^T u, \quad (21)$$

where $e_\star = \frac{1}{2} x_\star^H W_\star x_\star$, $\phi_\star(x_\star) = \Re(B_\star^H W_\star x_\star)$ and $\psi_\star(x_\star) = x_\star^H Q_\star x_\star$ with $Q_\star = -\Re(W_\star A_\star)$, so that the two following key points are fulfilled:

- (i) e_\star is a positive definite function of x_\star (since W_\star is positive definite);
- (ii) Q_\star is positive definite if and only if all the eigenvalues σ (the poles of the transfer matrix) have a strictly negative real part.

Step 4. Write these results in the original state basis: (10) is recovered with $e_\star = \mathcal{V}(x)$, $\frac{1}{\omega} \mathcal{P}(x, u) = -\tilde{\psi}(x) + \tilde{\phi}(x)^T u$, and

$$\mathcal{V}(x) = \frac{1}{2} x^T W x \quad \text{with } W = W^H = P^{-H} W_\star P^{-1}, \quad (22)$$

$$\tilde{\psi}(x) = x^T Q x \quad \text{with } Q = Q^H = -P^{-H} \Re(W_\star A_\star) P^{-1}, \quad (23)$$

$$\tilde{\phi}(x) = L x \quad \text{with } L = B^T P^{-H} W_\star P^{-1} = B^T W, \quad (24)$$

for any $W_\star = \text{diag}(w_\star) > 0$ and where $P^{-H} = (P^{-1})^H$.

This method builds a family of “candidate” Lyapunov functions parametrized by a definite positive diagonal weight matrix $W_\star = \text{diag}(w_\star)$.

Remark 4 (Time-invariant parameter r). *The validity of (21) is conditioned by the fact that w_\star does not depend on time and that of step 4 by the fact that P does not depend on time. Actually, P (related to A) depends on r (but not ω) so that the method gives “candidate” Lyapunov functions for time-invariant r and possibly time-varying ω .*

4.3.2. Application

In step 1, the computation of eigenvectors leads to

$$v_1 = \left[\sqrt[4]{r^3}, +\frac{1-i}{2}\sqrt{r}, -\frac{i}{2}\sqrt[4]{r}, -\frac{1+i}{4} \right]^T, \quad \text{for } \sigma = \sigma_1,$$

$$v_2 = \left[\sqrt[4]{r^3}, -\frac{1+i}{2}\sqrt{r}, +\frac{i}{2}\sqrt[4]{r}, +\frac{1-i}{4} \right]^T, \quad \text{for } \sigma = \sigma_2,$$

and $P^{-1} = [v_1, \overline{v_1}, v_2, \overline{v_2}]^{-1}$ is given by

$$P^{-1} = \begin{bmatrix} 1 & 1+i & i & -1+i \\ 1 & 1-i & -i & -1-i \\ 1 & -1+i & -i & 1+i \\ 1 & -1-i & i & 1-i \end{bmatrix} \left(\text{diag}[4\sqrt[4]{r^3}, 4\sqrt{r}, 2\sqrt[4]{r}, 2] \right)^{-1}.$$

Then, Choosing an uniform weight $w_\star = 4\sqrt{r^3} [1, 1, 1, 1]^T$, the results of step 4 leads to (22-24) with

$$W = \text{diag}(w), \quad \text{with } w = [1, 2\sqrt{r}, 4r, 8\sqrt{r^3}]^T,$$

$$Q = Q^T = \begin{bmatrix} 1 & -\sqrt{r} & 0 & 2r \\ * & 2\sqrt{r} & -2r & 0 \\ * & * & 4r & -4\sqrt{r^3} \\ * & * & * & 8\sqrt{r^3} \end{bmatrix},$$

$$L = [1, 0, 0, 0].$$

The sub-determinants (18) are given by

$$\begin{aligned} d_1(Q) &= 1, & d_2(Q) &= (2 - \sqrt{r})\sqrt{r}, \\ d_3(Q) &= 8\sqrt{r^3}(1 - \sqrt{r}) \quad \text{and} \quad d_4(Q) &= 64r^3(1 - \sqrt{r})^2. \end{aligned}$$

They are all strictly positive for $r \in]0, 1[$.

Hence, the maximal domain is restored, except the special case $r = 0$. Actually, in this case, the power balance based on eigenvectors is degenerated, since eigenvectors becomes all collinear and, eventually, do not take account of all capacitors anymore.

4.4. Results summary: Passivity and asymptotic stability of the linearized Moog filter

The previous studies (§ 4.2-4.3) allow to state the following result.

The linearized version of the Moog Ladder Filter (§ 4.1.1) is strictly passive and its equilibrium point $(x, u) = (0, 0)$ is asymptotically stable under one of the following conditions:

case a: for all time-varying parameters (ω, r) lying in

$$\mathcal{D}_{natural} = \mathbb{R}_+^* \times [0, \frac{5}{12}].$$

case b: for all time-varying parameter ω and time-invariant parameter r lying in

$$\mathcal{D}_{max} = \mathbb{R}_+^* \times [0, 1].$$

Moreover, the dynamics of the system fulfills strictly passive power balances w.r.t. the output $y = \omega \tilde{\phi}(x, u)$, based on adapted Lyapunov functions given by, for cases $\eta = a$ and $\eta = b$,

$$\mathcal{V}_{\ell in}^\eta(x) = w_\eta^T \underline{V}_{\ell in}(x) \quad (25)$$

with $\underline{V}_{\ell in}(x) = [V_{\ell in}(x_1), \dots, V_{\ell in}(x_4)]^T$, and $V_{\ell in}(x_k) = \frac{x_k^2}{2}$,

and (10) with $\frac{1}{\omega} \mathcal{P}_\eta(x, u) = -\tilde{\psi}_\eta(x) + \tilde{\phi}_\eta(x, u)^T u$

$$\tilde{\psi}_\eta(x) = x^T Q_\eta x \quad \text{with } Q_\eta = -\frac{1}{2} (W_\eta A + A^T W_\eta)$$

$$\tilde{\phi}_\eta(x, u) = L_\eta x \quad \text{with } L_\eta = B^T W_\eta$$

and where $\tilde{\psi}_\eta$ is positive definite choosing $W_\eta = \text{diag}(w_\eta)$ with

$$w_a = [1, 1, 1, 1]^T, \quad w_b = [1, 2\sqrt{r}, 4r, 8\sqrt{r^3}]^T, \quad (\text{if } r(t) = r(0) = r).$$

Remark 5 (Local dissipativity). Although the case $\eta = b$ gives a strong result only for a constant parameter r , the study developed in § 4.3 allows to state a weaker result, including for time-varying parameters lying in \mathcal{D}_{max} .

Let $r \in [0, 1[$ be a fixed value. Consider $\ell = a$ if $r = 0$, $\ell \in \{a, b\}$ if $0 < r < 5/12$ and $\ell = b$ if $5/12 \leq r < 1$. Moreover, denote $(\tilde{\omega}, \tilde{r})$ the time-varying parameters which monitor the linearized filter, lying in \mathcal{D}_{max} . If $\tilde{r}(t) = r$ at a time t , then the power balance (10) is fulfilled at time t .

5. NONLINEAR SYSTEM ANALYSIS

In this section, the passivity and the Lyapunov stability are examined for the nonlinear system. The study is based on a storage function which allows to obtain a formulation similar to that of the linearized system and to take benefit from the results stated in section 4. The derivations are presented in three steps. First, a remarkable identity on the feedback loop is exhibited. Second, the identity is used to reformulate the state-space equation (8) in a way similar to (12). Third, the storage function, the passivity and the Lyapunov analysis are presented.

5.1. Step 1: remarkable identity on the feedback loop

From $\tanh(a+b) = \frac{\tanh a + \tanh b}{1 + \tanh a \tanh b}$, we get $\tanh(u - 4rx_4) = -\tanh(4rx_4) + \frac{(1 - \tanh^2(4rx_4)) \tanh u}{1 - \tanh(4rx_4) \tanh u}$ which rewrites

$$\tanh(u - 4rx_4) = -4\rho_r(x_4) \tanh(x_4) + \beta(rx_4, u) u, \quad (26)$$

where functions ρ_r and β are positive regular functions.

The function $\rho_r : \mathbb{R} \rightarrow I_r = \rho_r(\mathbb{R})$ is defined by

$$\rho_r(x_4) = \frac{\tanh(4rx_4)}{4 \tanh x_4}, \quad \text{if } x_4 \neq 0, \text{ and } \rho_r(0) = r, \quad (27)$$

where $I_0 = \{0\}$, $I_r = [r, \frac{1}{4}]$ if $0 < r < \frac{1}{4}$, $I_{\frac{1}{4}} = \{\frac{1}{4}\}$ and $I_r =]\frac{1}{4}, r]$ otherwise (see Fig. 2 for an illustration). Hence, a property is that, for all $x_4 \in \mathbb{R}$, if $r \in [0, 1[$ then $\rho_r(x_4) \in [0, 1[$ as well.

The function $\beta : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is defined by

$$\beta(z, u) = \frac{1 - \tanh^2 z}{1 - \tanh u \tanh z} \mu(u), \quad (28)$$

$$\text{with } \mu(u) = \frac{\tanh u}{u}, \quad \text{if } u \neq 0, \text{ and } \mu(0) = 1. \quad (29)$$

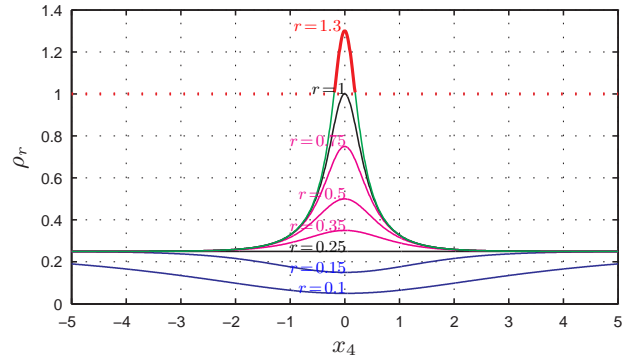


Figure 2: Function $x_4 \mapsto \rho_r(x_4)$ for several positive values of r . This function plays the same role (a feedback gain) as r in the linearized system, for the nonlinear system. For $r = 1.3$, the red part exceeds the stability limit gain value 1.

Note that in (28), the first factor is positive, finite and regular (but not bounded), and that μ is positive, finite, regular and lower than 1 (see Fig. 3 for an illustration), so that function β is well-posed.

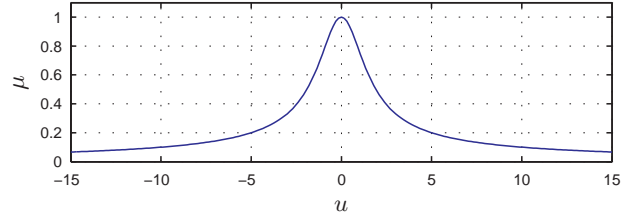


Figure 3: Function $u \mapsto \mu(u)$ defined in (29).

5.2. Step 2: State-space formulation similar to (12)

Using identity (26) in (8) and introducing

$$\Theta(x) = [\tanh x_1, \tanh x_2, \tanh x_3, \tanh x_4]^T, \quad (30)$$

lead to the state-space equation

$$\frac{1}{\omega} \frac{dx}{dt} = \mathcal{A} \Theta(x) + \mathcal{B} u \quad (31)$$

where the matrices \mathcal{A} and \mathcal{B} are functions of, respectively, $\rho_r(x_4)$ and $\beta(rx_4, u)$, which are given by

$$\mathcal{A} = \begin{bmatrix} -1 & 0 & 0 & -4\rho_r(x_4) \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad \text{and} \quad \mathcal{B} = \begin{bmatrix} \beta(x, u) \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (32)$$

Remark 6 (Gains). $\rho_r(x_4)$ can be interpreted as a feedback gain and $L(rx_4, u)$ as the input gain.

5.3. Step 3: Storage function, passivity and Lyapunov analysis

Let $w \in (\mathbb{R}_+^*)^4$, $W = \text{diag}(w)$ and consider the storage function $\mathcal{V}_{n\ell}$ defined as (25) where $V_{\ell in}$ is replaced by (see Fig. 4)

$$\forall x_k \in \mathbb{R}, \quad V_{n\ell}(x_k) = \ln \cosh(x_k). \quad (33)$$

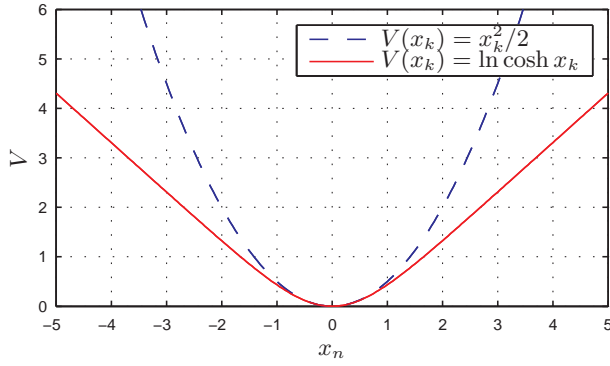


Figure 4: Storage functions: V_{lin} (--) defined by (16) is used for the linearized system analysis and V_{nl} (-) (33) for the nonlinear system analysis.

Remark 7 (Function V_{nl}). The storage function V_{nl} defined by (33) does not longer correspond to the energy stored in a capacitor, except for small signals since $\ln \cosh x \sim \frac{1}{2}x^2$ (see also Fig. 4).

Then, following (31), and as $V'_{nl}(x_k) = \tanh x_k$,

$$\begin{aligned} \frac{1}{\omega} \frac{dV_{nl}(x)}{dt} &= (\Theta(x)^T W) f(x, u) \\ &= \Theta(x)^T W \mathcal{A} \Theta(x) + \Theta(x)^T W \mathcal{B} u, \end{aligned}$$

so that $\frac{P_{nl}(x)}{\omega} = -\tilde{\psi}_{nl}(x) + \tilde{\phi}_{nl}(x, u) u$ where

$$\begin{aligned} \tilde{\psi}_{nl}(x) &= \Theta(x)^T Q_{nl}(\rho_r) \Theta(x) \quad \text{with } Q_{nl} = -\frac{1}{2}(W \mathcal{A} + \mathcal{A}^T W), \\ \tilde{\phi}_{nl}(x, u) &= L_{nl}(x, u) \Theta(x) \quad \text{with } L_{nl} = \mathcal{B}^T W. \end{aligned}$$

As $\tilde{\psi}_{nl}$ is a quadratic form w.r.t. $\Theta(x)$, it is positive definite iff Q_{nl} is a positive definite matrix as well. This property is achieved as in section 4 for $w = w_a$ since \mathcal{A} is the same matrix as A in which $r \in [0, 5/12[$ has been replaced by $\rho_r(x_4) \in [0, 5/12[$.

As a consequence, the passivity w.r.t $y = \omega \tilde{\phi}_a$ and the Lyapunov stability are proved for any time-varying parameter lying in $\mathcal{D}_{natural}$ (§ 4.4-case $\eta = a$). Nevertheless, the results of (§ 4.4-case $\eta = b$) cannot be used here, but only the remark 5, since $\rho_r(x_4)$ varies with x_4 (even for fixed r).

5.3.1. Conclusion

Loosing the result of section 4.4 (case $\eta = b$) for the nonlinear system appears awkward. In practice, this is compensated by the fact that the feedback-loop gain ρ_r ($\forall r > 0$) becomes stabilizing again for large x_4 as soon as it falls under the critical value 1 (or $5/12$ for V_{nl} with the weight $w = [1, 1, 1, 1]$), as stated in the remark below. Finding a constant weight w restoring \mathcal{D}_{max} for the linear case would solve also the problem for the nonlinear case.

Remark 8 (Stability limit). The stability limit is reached when the gain ρ_r equals to 1, that is, only at $x_4 = 0$ in the case where $r = 1$. But, the more x_4 deviates from 0, the more $\rho_r(x_4)$ decreases (see Fig 2) and reinforces the stabilization. In this case, $x = 0$ is a

limit stable equilibrium point for $u = 0$. If $r \geq 1$, the system is locally unstable on $x_4 \in \{x_4 \mid \rho_r(x_4) > 1\} = I_r$ (see the red part in Fig. 2), but locally stabilized on $\mathbb{R} \setminus I_r$. Since $x = 0$ is the only equilibrium point for $u = 0$, the system can become self-oscillating.

6. NUMERICAL SCHEME AND DISSIPATIVE BEHAVIOUR

In signal processing, the bilinear transform is an extensively used numerical scheme. One reason is that it preserves the stability domain for time-invariant linear filters and usually lead to expected behaviours also for some time-varying and nonlinear cases. The question addressed here is to estimate how this numerical scheme, used here as in [2, 3], is able to fulfill a power balance close to a discrete-time (DT) version of (9-11).

For sake of conciseness, the notations used in the following part are $\frac{dx}{dt} = f(x, u, \alpha)$ where α denote the (possibly time-varying) parameters $\alpha = (\omega, r)$ and $x(n)$ denotes the variable at time $t = n\tau$ (rather than $x(n\tau)$) for the sampling frequency $F_s = 1/\tau$.

6.1. Bilinear transform

Applying the bilinear transform to (8) leads to

$$x(n+1) - x(n) = \frac{\tau}{2} \left[f(x(n+1), u(n+1), \alpha(n+1)) + f(x(n), u(n), \alpha(n)) \right]. \quad (34)$$

The simulation of the dynamics is processed by computing

$$x(n+1) = x(n) + \delta(n), \quad (35)$$

where $\delta(n)$ is governed by (34) in which $x(n+1)$ is replaced by (35). The computation of $\delta(n)$ is processed either by using a Newton-Raphson algorithm [14] or by approximating the solution by that of the first order Taylor expansion (w.r.t. $\delta(n)$) of its governing equation, that is,

$$\begin{aligned} \delta(n) &= \frac{\tau}{2} \left[I_4 - \frac{\tau}{2} \partial_{(x^T)} f(x(n), u(n+1), \alpha(n+1)) \right]^{-1} \\ &\quad \times \left[f(x(n), u(n), \alpha(n)) + f(x(n), u(n+1), \alpha(n+1)) \right]. \quad (36) \end{aligned}$$

In practice, the latter solution is accurate if the cutoff frequency is sufficiently low ($\omega/(2\pi) \ll F_s/2$). In this case, the computation cost can still be reduced without deteriorating the result by replacing the last factor of (36) by $f\left(x(n), \frac{u(n+1)+u(n)}{2}, \frac{\alpha(n+1)+\alpha(n)}{2}\right)$.

6.2. Discrete-time power balance and dissipated contribution

The passivity is examined for the storage function $\frac{dV_{nl}(x)}{dt} = w^T \frac{dV_{nl}(x)}{dt}$ with $V_{nl}(x) = [V_{nl}(x_1), \dots, V_{nl}(x_4)]^T$. Following (34-35), a DT version of $\frac{dV_{nl}(x_k)}{dt} = V'_{nl}(x_k) \frac{dx_k}{dt}$ is

$$\begin{aligned} \frac{V_{nl}(x_k(n+1)) - V_{nl}(x_k(n))}{\tau} &= \Delta V(x_k(n), \delta_k(n)) \frac{\delta_k(n)}{\tau}, \\ \Delta V(\xi, \delta) &= \frac{V_{nl}(\xi + \delta) - V_{nl}(\xi)}{\delta} \left(\xrightarrow{\delta \rightarrow 0} V'_{nl}(\xi) = \tanh \xi \right), \end{aligned}$$

where $\delta(n)$ is computed using (34-35).

To characterize the dissipativity in the sense of this DT power balance, the associated function ψ^d must be identified. This can be done remarking that, according to (10-11), functions $\psi(x, \alpha) = w^T \underline{\psi}(x)$ and $\phi(x, u, \alpha) = w^T \underline{\phi}(x, u, \alpha)$ are here formally computed with $\psi_k(x_k, \alpha) = -V'_{n\ell}(x_k) f(x, 0, \alpha)$ and $\phi_k(x, u, \alpha) = [V'_{n\ell}(x_k) f_k(x_k, u, \alpha) + \psi(x_k, \alpha)]/u$. The looked-for DT versions are then given by replacing $V'_{n\ell}$ by δV and occurrences of f by their corresponding averages at samples n and $n+1$. The DT dissipated contribution ψ^d computed in this way is given by

$$\psi^d(x(n), \delta(n); \alpha(n), \alpha(n+1)) = -\underline{\Delta V}(x(n), \delta(n))^T \times \text{diag}(w) \times \frac{f(x(n), 0, \alpha(n)) + f(x(n) + \delta(n), 0, \alpha(n+1))}{2}. \quad (37)$$

with $\underline{\Delta V}(x, \delta) = [\Delta V(x_1, \delta_1), \dots, \Delta V(x_4, \delta_4)]^T$ and makes sense. It yields a DT Lyapunov principle for $u = 0$ if $\psi^d \geq 0$: $V_{n\ell}(x_k(n+1)) - V_{n\ell}(x_k(n)) = -\tau \psi^d(x(n), \delta(n); \alpha(n), \alpha(n+1))$. The positivity domain of ψ^d is not straightforward to exhibit even for constant parameters (ω, r) : in this case, ψ can be written as a quadratic form w.r.t. $\theta(n) + \theta(n+1)$ but with a weight matrix which depends on the time, as stated in the following remark.

Remark 9. For constant parameters, ψ is a quadratic function w.r.t. $T(\xi, \delta) = (\tanh(\xi + \delta) + \tanh \xi)/2$. This is obtained rewriting $\Delta V(\xi, \delta) = F(\xi, \delta) T(\xi, \delta)$ where the introduced function F can be interpreted as a correction factor due to the time-discretization. This factor is proved to be larger than 1 and such that $F(\xi, \delta) \xrightarrow{\delta \rightarrow 0} 1$.

6.3. Simulations and results

Simulations presented below are performed using (36) with $F_s = 48\text{kHz}$. The input is a linear sweep $u(t) = a \sin \phi(t)$ with $\phi(t) = 2\pi(f^-t + \frac{f^+ - f^-}{2t^*}t^2)$ starting with frequency $f^- = 50\text{Hz}$ and ending with $f^+ = 2\text{kHz}$ at $t^* = 0.1\text{s}$. Three amplitudes are tested: $a = 0.01$ (very linear limit, except if the resonance is high), $a = 1$ (medium nonlinear dynamics) and $a = 5$ (highly nonlinear dynamics). The cutoff frequency is $f_c = 1\text{kHz}$.

For $r \leq 5/12$ (low resonance and limit of the proved passivity for the linear approximation of the filter, see § 4.2.2), the DT passivity is unconditionally satisfied ($\psi^d \geq 0$) for the indicator ψ^d built with the weight $w = w_a$ (energy stored in capacitors).

For $5/12 < r < 1$, this is no longer the case but the DT passivity is still mostly fulfilled for the indicator ψ^d built with the weight $w_b(r)$. The dissipativity violation can be appreciated in Fig. 5 for $r = 0.7$ (high resonance) and $r = 1.1$ (non asymptotically stable domain).

Note that, even in the latter case ($r > 1$), x_4 does not diverge. For the very low amplitude ($a = 0.01$), a self-oscillation appears, as for the analog circuit. For larger amplitudes, the filter is driven by the input so that no self-oscillation appear.

Hence, the bilinear transform has not proved to guarantee the passivity. However, this numerical study shows its relevance and its ability to capture some of the characteristic and expected behaviours of the Moog filter.

7. CONCLUSIONS AND PERSPECTIVES

In this paper, the passivity analysis of the Moog Ladder Filter circuit has been examined. Three families of candidate Lyapunov functions have been proposed for the linearized and the nonlinear versions of the system: in the linear case, the natural energy stored in capacitors and an adapted weighted sum of the energies conveyed by eigenvectors and, in the nonlinear case, an adaptation of elementary storage functions. The first one guarantees the passivity for any time-varying parameters on a restricted domain. The second one recovers the optimal stability domain for time-invariant feedback-loop gains in the linear case. The third one generalizes these results to the nonlinear case in an exact way for the first one, but it only gives some clues for the characterization of the optimal domain. Finally, these results have allowed to derive an dissipativity indicator for discrete-time simulations based on the bilinear transform.

The analysis of simulations reveals that the bilinear transform does not guarantee the dissipativity of the nonlinear filter if gains r are larger than $5/12$ (whether time-varying or not). However, even in this case, the dissipativity condition stays mostly fulfilled and simulations show that the bilinear transform generates the known characteristic behaviours of the Moog filter.

As a consequence, the main perspective of this study consists in deriving refined storage functions and a specific numerical scheme that (both) guarantee a discrete-time passivity. Conservative schemes (which preserve the energy of conservative systems and Hamiltonian systems) based on variational approaches have been developed [15] and are still an active field of research. They can reveal to be relevant also when they are applied to lossy versions of originally non-lossy problems (see e.g. [9, Apdx.A1]). Another perspective is concerned with the derivation of explicit schemes preserving the passivity. Finally, a deeper study on the time variation effects of parameter $r > 5/12$ should be carried on.

8. REFERENCES

- [1] R. A. Moog, "A voltage-controlled low-pass high-pass filter for audio signal processing," in *Proc. of the 17th AES Convention*, New York, 1965, pp. 1–12.
- [2] T. Stilson and J. Smith, "Analyzing the Moog VCF with considerations for digital implementation," in *Proc. of Int. Computer Music Conf.*, Hong Kong, China, 1996, pp. 398–401.
- [3] A. Huovilainen, "Non-linear digital implementation of the Moog ladder filter," in *Proc. of the Int. Conf. on Digital Audio Effects*, Naples, Italy, 2004, pp. 61–64.
- [4] T. E. Stinchcombe, "Derivation of the transfer function of the Moog ladder filter," Tech. Rep., http://mysite.wanadoo-members.co.uk/tstinchcombe/synth/Moog_ladder_tf.pdf, 2005.
- [5] J. Smith, "Virtual acoustic musical instruments: Review of models and selected research," in *Proc. Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE, 2005.
- [6] V. Valimaki and A. Huovilainen, "Oscillator and filter algorithms for virtual analog synthesis," *Computer Music Journal*, vol. 30, no. 2, pp. 19–31, 2006.

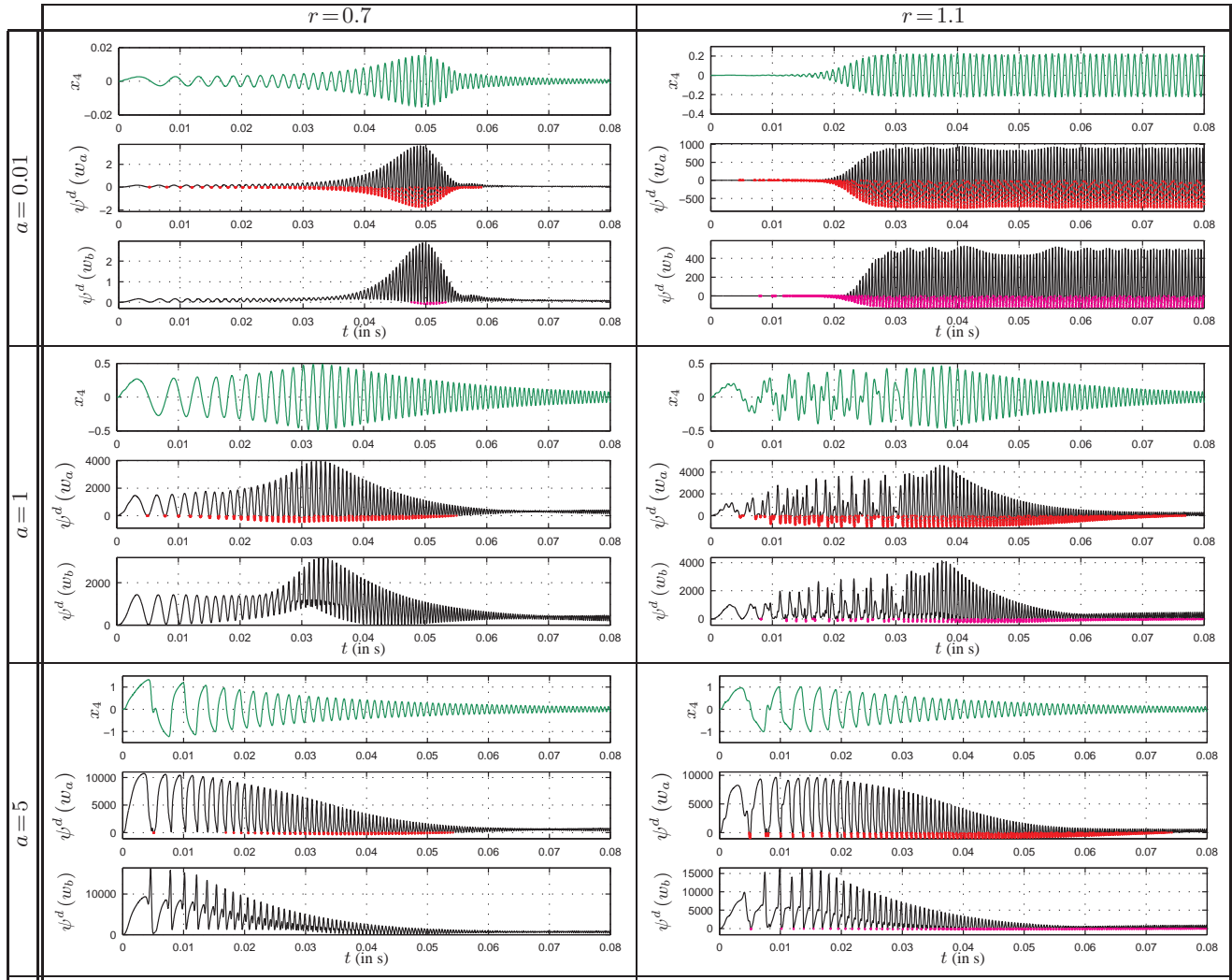


Figure 5: Compilation of simulations of x_4 (filter output) and of ψ^d with the weights w_a and $w_b(r)$. The dissipativity violation corresponds to $\psi^d < 0$ (associated samples are marked in red for w_a and in magenta for w_b). As expected, there are very few marked samples for $r = 0.7$. The time at which $f_{sweep}(t) = f_c$ is $t = 48.7$ ms.

- [7] T. Hélie, “On the use of Volterra series for real-time simulations of weakly nonlinear analog audio devices: application to the Moog ladder filter,” in *DAFx*, Montréal, Québec, 2006, vol. 9, pp. 7–12.
- [8] T. Hélie, “Volterra series and state transformation for real-time simulations of audio devices including saturations: application to the moog ladder filter,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18-4, no. 4, pp. 747–759, 2010.
- [9] S. Bilbao and J. Smith, “Energy-conserving finite difference schemes for nonlinear strings,” *Acta Acustica*, vol. 91, pp. 299–311, 2005.
- [10] S. Bilbao, “A family of conservative finite difference schemes for the dynamical von karman plate equations,” *Numerical Methods for Partial Differential Equations*, vol. 24, no. 1, pp. 193–216, 2008.
- [11] J. Chabassier and P. Joly, “Energy preserving schemes for nonlinear hamiltonian systems of wave equations. application to the vibrating piano string,” *Computer Methods in Applied Mechanics and Engineering*, pp. 2779–2795, 2010.
- [12] F. Fontana and M. Civolani, “Modeling of the EMS VCS3 voltage-controlled filter as a nonlinear filter network,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 760–772, 2010.
- [13] H. K. Khalil, *Nonlinear systems*, Prentice Hall, Upper Saddle River, NJ 07458, 3rd ed. edition, 2002.
- [14] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag Inc.; Édition : 3rd Revised edition, New York, 3rd edition, 2002.
- [15] E. Hairer, C. Lubich, and G. Warner, *Geometric Numerical Integration*, Springer, 2002.

A PRELIMINARY STUDY ON SOUND DELIVERY METHODS FOR FOOTSTEP SOUNDS

Luca Turchet, *

Medialogy Section
Department of Architecture, Design
and Media Technology
Aalborg University Copenhagen
Lautrupvang 15, 2750 Ballerup, DK
tur@create.aau.dk

Stefania Serafin,

Medialogy Section
Department of Architecture, Design
and Media Technology
Aalborg University Copenhagen
Lautrupvang 15, 2750 Ballerup, DK
sts@create.aau.dk

ABSTRACT

In this paper, we describe a sound delivery method for footstep sounds, investigating whether subjects prefer static rendering versus dynamic. In this case, dynamic means that the sound delivery method simulates footsteps following the subject. An experiment was run in order to assess subjects' preferences regarding the sound delivery methods. Results show that static rendering is not significantly preferred to dynamic rendering, but subjects disliked rendering where footstep sounds followed a trajectory different from the one they were walking along.

1. INTRODUCTION

Procedural sound synthesis is becoming a successful approach to simulate interactive sounds in virtual environments and computer games [1, 2]. One important category of sounds produced by action of subjects navigating in an environment is the sound of footsteps.

Recently, several algorithms have been proposed to simulate walking sounds. One of the pioneers in this field is Perry Cook, who proposed a collection of physically informed stochastic models (PhiSM) simulating several everyday sonic events [3]. Among such algorithms the sounds of people walking on different surfaces were simulated [4]. A similar algorithm was also proposed in [5], where physically informed models simulate several stochastic surfaces.

Recently, in [6] a solution based on granular synthesis was proposed. The characteristic events of footstep sounds were reproduced by simulating the so-called ground reaction force, i.e., the reaction force supplied by the ground at every step.

The research just described does not take into consideration the ability of footstep sounds to be rendered in a 3D space. Sound rendering for virtual environments has reached a level of sophistication that it is possible to render in realtime most of the phenomena which appear in the real world [7].

In this study, we are interested in investigating how subjects react to different kinds of sound rendering algorithms, which follow the user or propose different confusing trajectories.

The results presented in this paper are part of the Natural Interactive Walking (NIW) FET-Open project¹, whose goal is to provide closed-loop interaction paradigms enabling the transfer of skills that have been previously learned in everyday tasks associated to walking. In the NIW project, several walking scenarios

are simulated in a multimodal context, where especially audition and haptic play an important role.

2. SYSTEM ARCHITECTURE

The system we adopted for the experiments consists of a motion capture system (MoCap)², a soundcard³, eight loudspeakers, two sandals with pressure sensors embedded in, and two computers. Figure 1 shows a schematic representation of the overall architecture developed.

Such system was placed in an acoustically isolated laboratory which consisted of a control room and a bigger room where the setup was installed and where the experiments were performed. The control room was 5.45 m large, 2 m long, and 2.85 m high, and it was used by the experimenters providing the stimuli and collecting the experimental results. It hosted two desktop computers.

The first computer run the motion capture software, while the second run the footstep sounds synthesis engine (see section 2.1). The two computers were connected through an ethernet cable and communicate by means of the UDP protocol. The data relative to the motion capture system were sent from the first to the second computer which processed them in order to control the sound engine.

The experiment room was 5.45 m large, 5.55 m long, and 2.85 m high. A transparent glass divided the two rooms, so it was possible for the experimenters to see the users performing the assigned task. The two rooms were connected by means of a talkback system.

The user locomotion was tracked by an Optitrack motion capture system⁴, composed by 16 infrared cameras⁵. The cameras were placed in a configuration optimized for the tracking of the head position. In order to achieve this goal, markers were placed on the top of the head using a bicycle helmet. The walking area available to the users for the purposes of the experiments consisted of a rectangle 2.5 x 2.6 m which corresponded to the area fully seen by the infrared cameras. The perimeter of such rectangle was indicated on the floor by means of scotch tape strips (see figure 2)

Users were also tracked by using the pressure sensors embedded in a pair of shoes. Specifically, a pair of light-weight san-

* This work was supported by the NIW project

¹<http://www.niwproject.eu/>

²from Naturalpoint with software Tracking Tools 2.0

³FireFace 800 soundcard:

http://www.rme-audio.de/en_products_fireface_800.php

⁴<http://naturalpoint.com/optitrack/>

⁵OptiTrack FLEX:V100R2

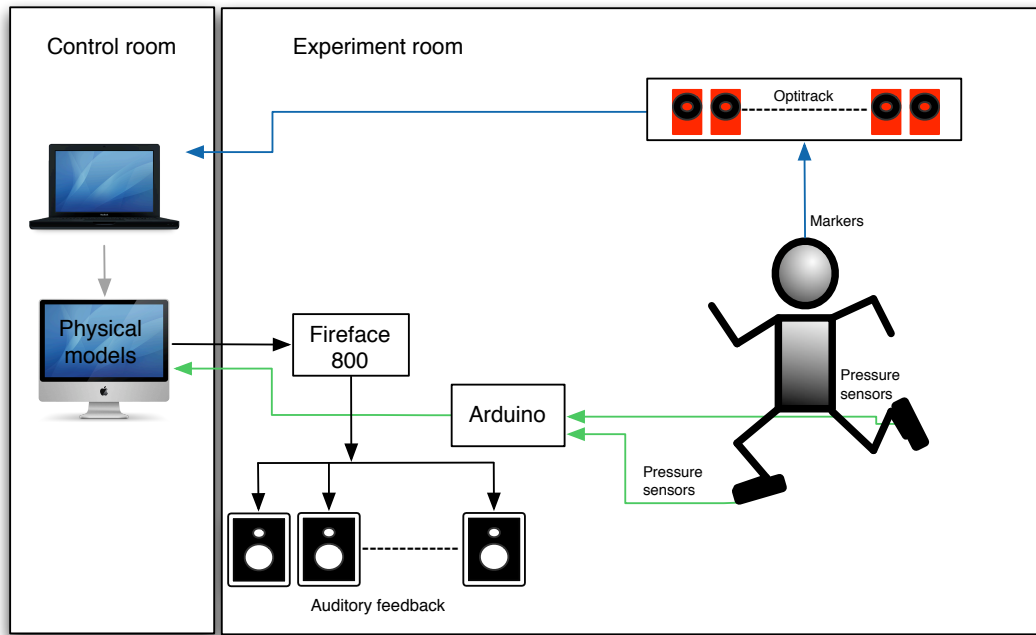


Figure 1: A block diagram of the architecture.

dals was used (Model Arpenaz-50, Decathlon, Villeneuve d'Ascq, France). The sole had two FSR pressure sensors⁶ whose aim was to detect the pressure force of the feet during the locomotion of a subject wearing the shoes. The two sensors were placed in correspondence to the heel and toe respectively in each shoe. The analogue values of each of these sensors were digitalized by means of an Arduino Diecimila board⁷ and were used to drive the audio synthesis.

The configuration of the eight loudspeakers is illustrated in Figure 2. In detail, the loudspeakers⁸ were placed on the ground at the vertices and at the middle point of the sides of the rectangular floor. During the experiments the loudspeakers were hidden from view using acoustically transparent curtains.

2.1. Footstep sounds synthesis engine

In previous research, we proposed a sound synthesis engine able to simulate footstep sounds on aggregate and solid surfaces [8]. Such engine is based on physical models which are driven by a signal, in the audio domain, expressing the ground reaction force (GRF), i.e., the reaction force supplied by the ground at every step. In our simulations the GRF corresponds to the amplitude envelope extracted from an audio signal containing a footstep sound.

The engine can operate both offline and in real-time. The two approaches differ in the way the input GRF is generated. Concerning the realtime work, various systems for the generation of such input have been developed and tested [8, 9, 10]. In the proposed experiments, the footstep sounds synthesis is driven interactively during the locomotion of the subject wearing the shoes. The description of the control algorithms based on the analysis of the

values of the pressure sensors coming from the shoes can be found in [11]).

The sound synthesis algorithms were implemented in C++ as external libraries for the Max/MSP⁹ sound synthesis and multimedia real-time platform.

2.2. Sound delivery methods

We implemented and tested two different types of approaches for the delivery of the footstep sounds through the loudspeakers: static and dynamic diffusion.

For static diffusion we intend that the footstep sound, generated interactively during the locomotion of the user wearing the shoes, is diffused simultaneously to the eight loudspeakers, and with the same amplitude in each loudspeaker.

Conversely, during the dynamic diffusion the user position was tracked by the MoCap and it was used to diffuse the footstep sound according to a sound diffusion algorithm based on ambisonics. Specifically, to achieve the dynamism we used the ambisonic tools for Max/MSP¹⁰ which allow to move virtual sound sources along trajectories defined on a tridimensional space [12]. Such algorithm was set in order to place under the user feet the virtual sound source containing the footstep sounds. In this way the sound followed the user trajectories during his/her locomotion, and therefore the eight loudspeakers delivered the footstep sounds with different amplitudes. As an example, in reference to figure 2, when the user position was near the loudspeakers 1 and 2, the effect resulting from the dynamic diffusion was that the sound was mostly delivered through these two loudspeakers while the loudspeakers 5 and 6, placed on the opposite sides, did not deliver any sound.

⁶I.E.E. SS-U-N-S-00039

⁷<http://arduino.cc/>

⁸Dynaudio BMSA speakers: <http://www.dynaudioacoustics.com/>

⁹<http://cycling74.com/>

¹⁰Available at <http://www.icst.net/research/projects/ambisonics-tools/>

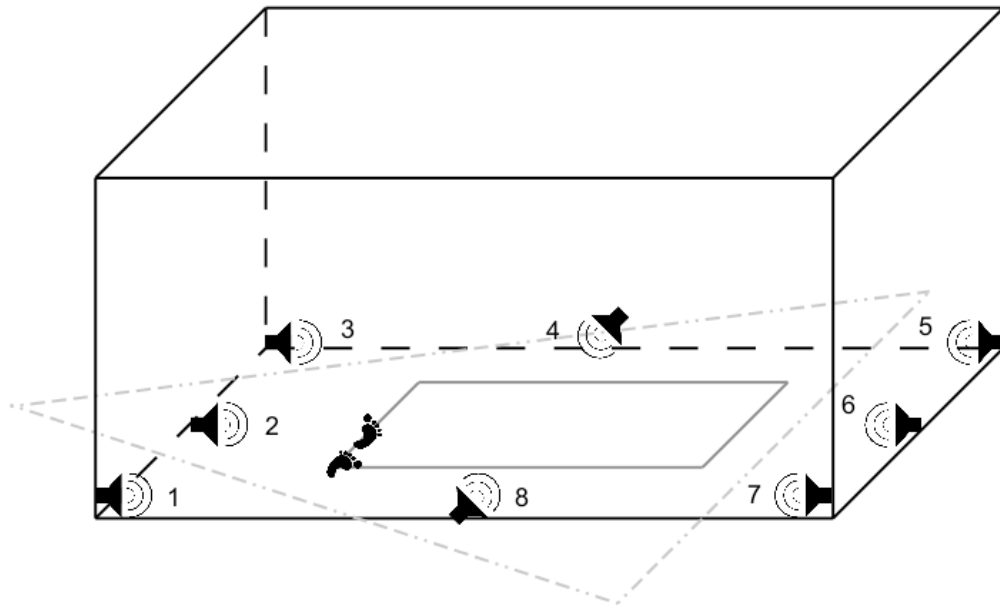


Figure 2: Loudspeakers configuration used in this study. In dark grey the the perimeter of the rectangle delimiting the walking area (indicated on the floor by means of scotch tape strips). In light grey the triangular trajectory for the delivery of the distractors during the dynamic diffusion condition in experiment 1.

In addition to the static and dynamic diffusion, during the experiments we also delivered the sounds using a second type of dynamic diffusion where the sound delivery was incoherent with the user position. In detail, in this configuration we chose to deliver the sound as coming from the side opposite to the user position. As an example, in reference to figure 2, when the user position was near the loudspeakers 1 and 2, the effect resulting from the dynamic diffusion was that the sound was mostly delivered through loudspeakers 5 and 6, while the loudspeakers 1 and 2 did not deliver any sound.

Finally, in presence of the dynamic diffusion, the delay due to the MoCap was negligible for the purposes of the experiments.

3. DESCRIPTION OF THE EXPERIMENTS

We performed two experiments in order to assess subjects' reaction to the sound delivery methods. During the experiments participants were asked to wear the sandals, and the cycling helmet mentioned in section 2 and walk in the laboratory according to the tasks of the two experiments.

3.1. Experiment 1

The task of the first experiment consisted on walking in circular way along the perimeter of the walking area (i.e. the rectangle indicated on the floor by means of scotch tape strips). During their walk they produced interactively footstep sounds which were delivered through the loudspeakers according to the following six conditions:

- static diffusion
- coherent dynamic diffusion
- incoherent dynamic diffusion

- static diffusion plus static distractors
- coherent dynamic diffusion plus dynamic distractors
- incoherent dynamic diffusion plus dynamic distractors

The three methods explained in section 2.2 were presented with and without distractors. Such distractors consisted of footstep sounds of a virtual person walking in the same room. Specifically, in presence of static diffusion the distractors were delivered statically (i.e., with the same volume in all the loudspeakers), while in the dynamic diffusion condition they were delivered dynamically following a triangular trajectory (see figure 2).

Participants were exposed to twelve trials, where the six conditions were presented twice in randomized order. Each trial lasted one minute.

The sound engine was set in order to synthesize footstep sounds on two different kinds of materials: wood and forest underbrush. Each condition was presented with both wood and forest underbrush. The reason for choosing two materials was to assess whether the surface type affected the quality of the results. In this particular situation, a solid and an aggregate surface were chosen.

The distractors were presented using the same surface chosen for the participants' walks. In order to keep the distinction between distractors and participants' footstep sounds simple, distractors were presented with a lower volume, a small change in the timber, and with a moderately quick gait.

After the presentation of each stimulus participants were required to evaluate on a seven-point Likert scale the following questions:

- How well could you localize under your feet the footstep sounds you produced?
- How well did the sounds of your footsteps follow your position in the room?

- How much did your walk in the virtual environment seem consistent with your walk in the real world?
- How natural did your interaction with the environment seem?
- To what degree did you feel confused or disoriented while walking?

The goal of this experiment was to compare the proposed diffusion methods. The incoherent dynamic diffusion was included in the experiment in order to assess if any difference in the participants evaluations between coherent dynamic and static diffusion, was not due only to the fact that the source was moving, but that it was moving coherently with the user position. In order to make a consistent comparison, the sound synthesis engine was set with appropriate volumes for footstep sounds delivered through the static and the dynamic diffusion, i.e. there was no big volume difference between the two sound delivery methods. The distractors were used to assess if the same differences in the participants evaluations of the three diffusion methods were found both in presence and in absence of distractors.

Our hypotheses were that the coherent dynamic condition would have got better results rather than the others (in particular the static one), that the incoherent dynamic condition would have been evaluated as the worst, and that the use of distractors would have worsened the participants evaluations in comparison with the case in which the distractors were not presented.

3.2. Experiment 2

Starting from the results of the first experiment we designed a second experiment in order to investigate in a deeper way the users' perception of the static and coherent dynamic diffusions. The task of such experiment consisted on walking freely inside the walking area. Participants were exposed to fourteen trials, where seven surface materials were presented in randomized order according to the two delivery methods (static and coherent dynamic diffusion). The seven surface materials, five aggregate and two solid, were gravel, sand, snow, dry leaves, forest underbrush, wood and metal. Each trial lasted one minute. After the presentation of each stimulus participants were required to evaluate on a seven-point Likert scale the same questions presented in the first experiment.

The goal of this experiment was to assess whether participants showed a preference for one of the two proposed methods when exploring the virtual environment by walking freely (and a without predefined trajectory like in experiment 1). Furthermore we were interested in assessing whether the type of used surface could affect the quality of the results.

3.3. Results of experiment 1

The first experiment was performed by thirteen subjects, 10 males and 3 females, aged between 21 and 38 (mean=24, standard deviation=4.51), who took on average about 17 minutes to complete it. Results are illustrated in figure 3. Various ANOVA (with and without repeated measures) were performed in order to assess if the differences found in the results were significative. All post-hoc analyses were performed using the least significant difference (LSD) test with Bonferroni's correction.

The first noticeable thing is the difference between results of the two surface materials, wood and forest underbrush, for what concerns the dynamic coherent condition and its comparison with the

static condition, in the case of absence of distractors. Indeed while for the wood material the differences between such two conditions are negligible for all the investigated parameters, instead for the forest underbrush material the differences are noticeable and significant (p -value = 0.000064). In presence of distractors such behavior is not hold neither for wood nor for forest, and all the differences are not significative.

A trend common to the two materials is that always the dynamic incoherent condition, both in presence and in absence of distractors, gave rise to lower evaluations in terms of localization, following, consistency and naturalness, and higher evaluations for what concerns the disorientation. In detail, for both materials, significance has been found concerning the differences between the dynamic coherent and dynamic incoherent conditions (for wood: p -value = 0.000285 and p -value = 0.005136, for forest: p -value < 0.000001 and p -value = 0.043566, for the cases with and without distractors respectively), and between the static and dynamic incoherent conditions (for wood: p -value = 0.004283 and p -value = 0.002424, for forest: p -value < 0.000001 both for the cases with and without distractors).

For both materials, as regards the parameters localization, following, consistency and naturalness, almost always the evaluations in absence of distractors are higher than when the distractors are present (the opposite behavior coherently happens for what concerns the disorientation parameter). This is more evident for the forest underbrush material, and indeed the difference between these two conditions is significant (p -value = 0.03817), while for wood is not.

In particular for both materials the disorientation is higher in presence of distractors rather than in absence, but significant difference between these two conditions was found only for the forest underbrush material (p -value = 0.01923). The condition dynamic incoherent with distractors was evaluated as the most disorienting in both materials; conversely, for the forest underbrush material only, the dynamic coherent condition was evaluated as the less disorienting.

As previously said, at global level the dynamic coherent condition gave rise to significant better results than the static one for what concerns the forest material in absence of distractors. In addition a successive analysis for each of the investigated parameters revealed significant difference between the two conditions only for the naturalness parameter (p -value = 0.013238).

3.4. Results of experiment 2

The second experiment was performed by ten subjects, 8 males and 2 females, aged between 19 and 37 (mean=28.8, standard deviation=5.63), who took on average about 17 minutes to complete it.

Results are illustrated in figure 4. As it is possible to notice, participants did not show any preference for one of the two methods. Evaluations of the investigated items of the questionnaire were very similar between the two methods for all the surfaces (all the differences are not statistically significant).

However it is possible to notice that the participant answers to the questionnaire items were not always similar for each surface material. In particular it is possible to observe that the metal surface on average produced the lower scores for the localization, naturalness and consistency items, and the higher scores for the disorientation item.

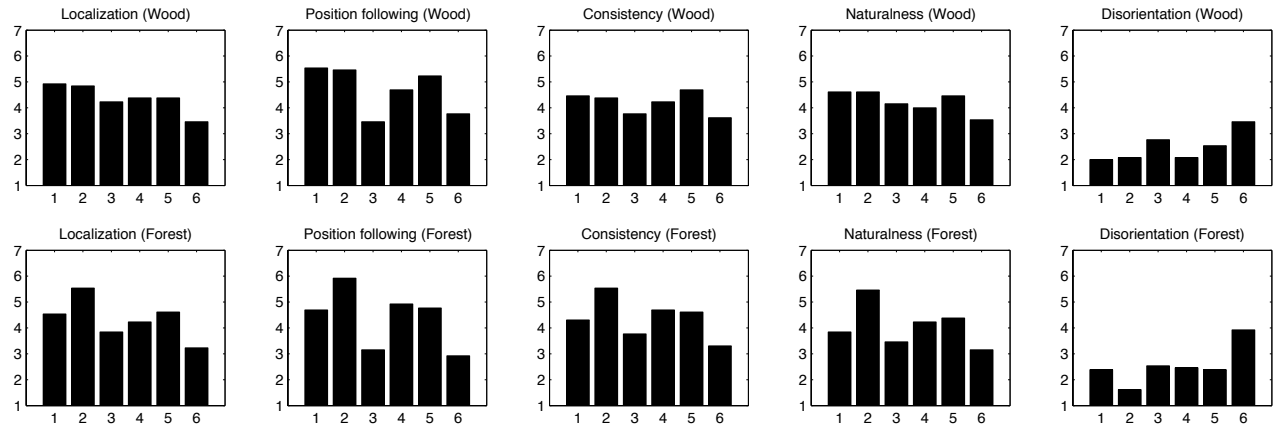


Figure 3: Results of the first experiment. Conditions are indicated on x-axis: 1- static diffusion, 2- coherent dynamic diffusion, 3- incoherent dynamic diffusion, 4- static diffusion plus static distractors, 5- coherent dynamic diffusion plus dynamic distractors, 6- incoherent dynamic diffusion plus dynamic distractors.

4. GENERAL DISCUSSION

The first noticeable element emerging from the results of the first experiment is the different behavior found for the coherent dynamic condition in the two simulated materials, in the case of absence of distractors. For forest underbrush such condition seems to play an important role since it got the best evaluations among all the investigated parameters, while for wood it got evaluations very similar to those of the static condition. So our hypothesis that people preferred the dynamic coherent condition to the static one was only partially confirmed.

Instead the hypothesis that the dynamic incoherent condition would have given the worst evaluations was confirmed, and found statistically significant for both materials and both in presence and absence of distractors.

It is therefore possible to conclude that users can perceive very well that their interaction with the virtual environment is not realistic nor natural when the source is not moving coherently with their position. This is an indication of the success of our simulations. The hypothesis concerning the distractors was confirmed: for both materials, almost always the evaluations in absence of distractors are better than when the distractors are present, although significant differences were found only for forest. In addition, the evaluations of the disorientation parameter were higher in presence of distractors (but significant only for forest). This indicates that the use of distractors, i.e., walking sounds evoking the presence of another person walking in the same room as the subject, is likely to influence the perception of footstep sounds associated with the subject.

Concerning the second experiment, results were clear: participants' evaluations did not differ for the two proposed methods, and this is an indication that the two methods could both be used in a virtual environment to deliver interactively generated footsteps sounds. However, other tests should be conducted in order to confirm this and assess more in detail other eventual differences in the perception of the two methods.

5. CONCLUSION AND FUTURE WORK

In this paper we have described an experiment whose goal was to assess the importance of surround sound rendering in simulating footstep sounds for virtual environments. Results show that static delivery method is not significantly preferred to the (coherent) dynamic one, and that participants disliked the renderings where footstep sounds followed a trajectory different from the one they were walking along.

In future experiments we will investigate in a deeper way the differences between the static and dynamic diffusion methods, as well as other parameters related to sound rendering, such as the role of reverberation and the role of amplitude.

We also plan to integrate the proposed footstep sounds renderings in an audio-haptic-visual environment, to design and evaluate different multimodal experiences based on walking.

6. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme under FET-Open grant agreement 222107 NIW - Natural Interactive Walking.¹¹

7. REFERENCES

- [1] P.R. Cook, *Real sound synthesis for interactive applications*, AK Peters, Ltd., 2002.
- [2] A. Farnell, *Designing Sound*, MIT Press, 2010.
- [3] P.R. Cook, "Physically Informed Sonic Modeling (PhISM): Synthesis of Percussive Sounds," *Computer Music Journal*, vol. 21, no. 3, pp. 38–49, 1997.
- [4] P. Cook, "Modeling Bill's Gait: Analysis and Parametric Synthesis of Walking Sounds," *Proceedings of the AES 22nd International Conference on Virtual, Synthetic, and Entertainment Audio*, pp. 73–78, 2002.

¹¹Natural Interactive Walking Project: www.niwproject.eu

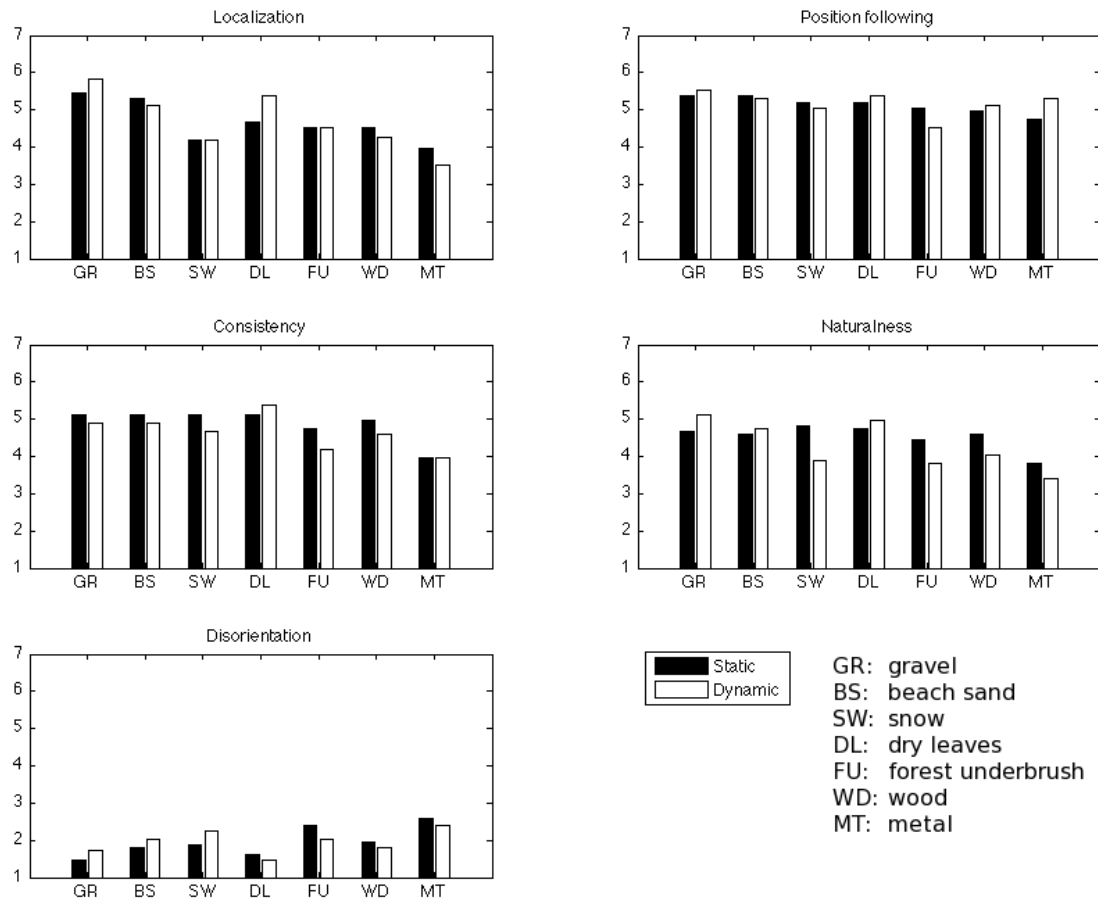


Figure 4: Results of the second experiment.

- [5] F. Fontana and R. Bresin, "Physics-based sound synthesis and control: crushing, walking and running by crumpling sounds," *Proc. Colloquium on Musical Informatics*, pp. 109–114, 2003.
- [6] A.J. Farnell, "Marching onwards: procedural synthetic footsteps for video games and animation," *Proceedings of the Pure Data Convention*, 2007.
- [7] T. Funkhouser, N. Tsingos, and J.M. Jot, "Survey of methods for modeling sound propagation in interactive virtual environment systems," 2003.
- [8] L. Turchet, S. Serafin, S. Dimitrov, and R. Nordahl, "Physically based sound synthesis and control of footsteps sounds," in *Proceedings of Digital Audio Effects Conference*, 2010.
- [9] R. Nordahl, S. Serafin, and L. Turchet, "Sound synthesis and evaluation of interactive footsteps for virtual reality applications," in *Proc. IEEE VR 2010*, 2010.
- [10] L. Turchet, R. Nordahl, and S. Serafin, "Examining the role of context in the recognition of walking sounds.," in *Proc. of Sound and Music Computing Conference*, 2010.
- [11] L. Turchet, R. Nordahl, A. Berrezag, S. Dimitrov, V. Hayward, and S. Serafin, "Audio-haptic physically based simulation of walking sounds.," in *Proc. of IEEE International Workshop on Multimedia Signal Processing*, 2010.
- [12] J.C. Schacher and M. Neukom, "Ambisonics Spatialization Tools for Max/MSP," in *Proceedings of the International Computer Music Conference*, 2006.

SIMULATION OF A VACUUM-TUBE PUSH-PULL GUITAR POWER AMPLIFIER

Jaromir Macak

Dept. of Telecommunications,
FEEC, Brno University of Technology
Brno, Czech Republic
jaromir.macak@phd.feec.vutbr.cz

Jiri Schimmel

Dept. of Telecommunications,
FEEC, Brno University of Technology
Brno, Czech Republic
schimmel@feec.vutbr.cz

ABSTRACT

Power amplifiers play an important role in producing of guitar sound. Therefore, the modeling of guitar amplifiers must also include a power amplifier. In this paper, a push-pull guitar tube power amplifier, including an output transformer and influence of a loudspeaker, is simulated in different levels of complexity in order to find a simplified model of an amplifier with regards to accuracy and computational efficiency.

1. INTRODUCTION

Guitar power amplifiers have a big effect for guitar sound. Power vacuum-tubes have characteristic nonlinear distortion [1]. Furthermore, compared to semiconductor amplifiers, tube power amplifiers have different circuit topology that brings other effects, such as typical frequency response and signal compression when a large input signal is supplied [1]. Moreover, output transformers play an important role in the output signal generation. The output signal is distorted by nonlinear hysteresis of a transformer core and frequency response of the output transformer [2].

Two types of topologies are used in tube power amplifier construction. Single ended power amplifiers consist mainly of one power tube, an output transformer, and resistors and capacitors. They provide asymmetrical limiting of the output signal that creates foremost even order harmonic components in the output signal spectrum. Push-pull power amplifiers have more complex topology. They consist of a phase splitter, two or four power tubes that processes opposite half-waves of the signal and output transformer that sums contributions from opposite power tubes. The push-pull amplifiers offer a symmetrical transfer function that creates odd order harmonic components in the output signal spectrum [3].

A single-ended power amplifier is simulated in [4] using wave digital filters. The model of the power amplifier contains a triode, a linearized model of the output transformer and a linearized model of a loudspeaker recomputed to its electrical equivalent. This model was improved in [5] where the complex nonlinear WDF model of the output transformer is used. A single-ended power amplifier with a pentode tube is discussed in [6]. This model also contains a linearized model of the output transformer but the loudspeaker is replaced by a constant load. Nevertheless, guitar tube amplifiers mostly contain a push-pull power amplifiers with two or four tubes (pentodes EL34 or beam tetrodes 6L6) that work in class AB or B [2]. However, the circuit topology and the circuit equations are the same for both classes and they differ only in bias point. A simulation of a push-pull amplifier is described in [1]. The complete circuit is divided into a phase splitter, a pentode circuit and a feedback circuit and they are connected using a modified

blockwise method [1]. However, this simulation uses a model of the ideal output transformer and the constant load.

In this paper, the pentode circuit is simulated in different level of complexity. At first, the pentode circuit with an ideal transformer and a constant load is solved, then the circuit with an ideal transformer and a linearized model of a loudspeaker. Finally, the circuit with a nonlinear model of the output transformer is simulated.

2. BASE CIRCUIT AND DEVICE MODELS

A typical pentode push-pull power amplifier contains two or four pentodes, the output transformer with connected loudspeaker and some resistors and capacitors (see figure 1). The circuit schematic also contains input units built by resistors R_g , R_b , capacitors C_g and bias voltage V_b . The input units are independent from the rest of the circuit because there is no coupling via the tube's cathode in topology with fixed bias. Their simulation is described in [1], and therefore the simulation of this part is omitted and there is only a focus on the output part of the amplifier. The typical values for circuit elements are listed in table 1.

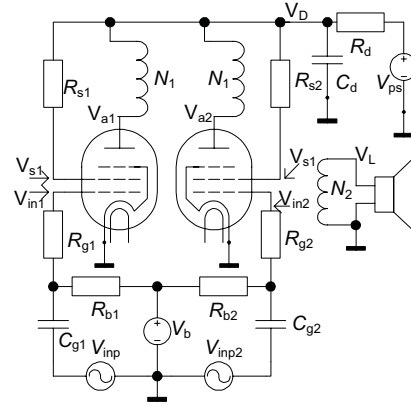


Figure 1: Circuit schematic of a push-pull tube amplifier.

Table 1: Values for circuit elements from figure 1.

$R_s [\Omega]$	$R_d [\Omega]$	$C_d [\mu F]$	N_1	N_2	$V_{ps} [V]$	$V_b [V]$
500	500	100	1560	60	500	-50

2.1. Pentode Model

The Koren's model of the pentode [7] was chosen as the pentode model. The pentode plate current is in form $I_a(V_{ak}, V_{gk1}, V_{gk2})$ where V_{ak} is plate-to-cathode voltage, V_{gk1} is the grid-to-cathode voltage and V_{gk2} is the screen-to-cathode voltage. The screen current is given in form $I_s(V_{gk1}, V_{gk2})$. The description of functions I_a and I_s is omitted here and is available in [7]. Frequency properties of the tube (e.g. Miller capacitance) are not considered because it should be included in the simulation of input unit.

2.2. Output Transformer Model

An ideal output transformer is considered to be an impedance divider that transforms input voltages V_p and currents I_p to output V_s and I_s according to

$$\frac{V_s}{V_p} = \frac{N_2}{N_1} = \frac{I_p}{I_s} \quad (1)$$

where N_1, N_2 are numbers of windings of the transformer. However, a real transformer is far away from the ideal one. For an accurate simulation, losses caused by hysteresis and core saturation have to be considered. Nonlinear behavior of the real transformer is described in numerous literature, e.g. [8, 9]. According to Ampere's law, the magnetizing force H is

$$H l_{\text{mag}} = N_1 I_p - N_2 I_s \quad (2)$$

where l_{mag} is the length of the induction path. The flux density B is computed from Faraday's law

$$\frac{\partial B}{\partial t} = \frac{V_s}{N_2 S} \quad (3)$$

where S is the transformer-core cross-section. The well-known nonlinear relation $B = \mu H$ can be implemented according to the Frolich equation [9] given by

$$B = \frac{H}{c + b|H|} \quad (4)$$

where c and b are constants derived from material properties. However, this model simulates only the core saturation. When simulating hysteresis, one can use e.g. Jiles-Atherton model [10] modified in [8] in order to remove nonphysical behavior of minor hysteresis loops. Magnetization of the core is obtained from

$$\frac{\partial M}{\partial H} = \delta_M \frac{M_{\text{an}} - M}{k\delta} + c \frac{\partial M_{\text{an}}}{\partial H} \quad (5)$$

where M_{an} is anhysteretic curve given by

$$M_{\text{an}} = M_s \left(\coth \left(\frac{H + \alpha M}{a} \right) - \frac{a}{H + \alpha M} \right) \quad (6)$$

and $\delta = \text{sign}(\partial H / \partial t)$. Parameters M_s, α, a, c and k are derived from material properties and their identification can be found e.g. in [11]. Parameter $\delta_M = 0$ when the nonphysical minor loop is going to be generated (anhysteretic magnetization has lower value than the irreversible magnetization) alternatively $\delta_M = 1$ [8]. Flux density is then obtained from

$$B = \mu_0 (M + H). \quad (7)$$

2.3. Loudspeaker Model

Loudspeakers play a very important role in the output signal generation via its frequency response. When considering linearized loudspeakers, one can model the frequency response with measured impulse responses with good results [1]. However, it is important to simulate the interaction between the tube amplifier and loudspeaker because the loudspeaker impedance is frequency dependent. The impedance can be modeled using the circuit schematic in figure 2 [12]. The values are derived from the added mass method and Thiele/Small parameters of a Celestion Vintage 30 loudspeaker placed in an Engl combo. The transformer leakage inductance and resistance can be modeled by modifying inductor L_{sp1} and resistor R_{sp1} values.

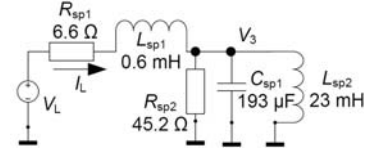


Figure 2: Simplified loudspeaker model – electric equivalent.

The loudspeaker impedance given by voltage V_L and current I_L can be expressed as the solution of the set of equations

$$\begin{aligned} I_L[n] &= I_L[n-1] + \frac{V_L[n] - I_L[n]R_{\text{sp1}} - V_3[n]}{L_{\text{sp1}}f_s} \\ V_3[n] &= V_3[n-1] + \frac{V_3[n]G_{\text{sp2}} - I_L[n] - I_{L2}[n]}{C_{\text{sp1}}f_s} \\ I_{L2}[n] &= I_{L2}[n-1] + \frac{V_3[n]}{L_{\text{sp2}}f_s} \end{aligned} \quad (8)$$

where $I_L[n-1], V_3[n-1], I_{L2}[n-1]$ are state variables and f_s is a sampling frequency. The equations were obtained using nodal analysis of the circuit in figure 2 and then discretized using Backward Euler formula.

Because the set of equations (8) is linear, it can be simplified into one linear equation

$$I_L[n] = -c_1 V_L[n] + I_{\text{tmp}} \quad (9)$$

where I_{tmp} is a linear combination of state variables given by

$$I_{\text{tmp}} = -c_2 I_L[n-1] + c_3 V_3[n-1] - c_4 I_{L2}[n-1]. \quad (10)$$

The new state variable values are then computed from

$$V_3[n] = -c_5 V_3[n-1] - c_6 I_L[n] + c_7 I_{L2}[n-1] \quad (11)$$

and

$$I_{L2}[n] = I_{L2}[n-1] + c_8 V_3[n]. \quad (12)$$

Coefficients c_{1-8} are derived from (8).

3. SIMULATION OF THE AMPLIFIER

In the simplest case, the load is considered to be constant. Using nodal analysis and discretization by Euler method, one can obtain the set of circuit equations

4. SIMULATION RESULTS

$$\begin{aligned}
0 &= -\frac{V_L[n]}{R_L} + \frac{N_1}{N_2} (I_{a1} - I_{a2}) \\
0 &= -V_{a1}[n] + V_D[n] - \frac{N_1}{N_2} V_L[n] \\
0 &= -V_{a2}[n] + V_D[n] + \frac{N_1}{N_2} V_L[n] \\
0 &= I_{s1} + \frac{V_D[n] - V_{s1}[n]}{R_{s1}} \\
0 &= I_{s2} + \frac{V_D[n] - V_{s2}[n]}{R_{s2}},
\end{aligned} \tag{13}$$

where V_D is voltage on the power supply, capacitor C_d is from the circuit schematic in figure 1 and $I_{a1} = I_a(V_{in1}[n], V_{a1}[n], V_{s1}[n])$, $I_s = I_s(V_{in1}[n], V_{s1}[n])$ and similarly $I_{s2} = I_s(V_{in2}[n], V_{s2}[n])$ and $I_{a2} = I_a(V_{in2}[n], V_{a2}[n], V_{s2}[n])$. The equations (13) are solvable for given value V_D and the solution can be implemented using a static waveshaper. It can be precomputed for different values of V_D voltage and stored in a look-up table. Then, during the simulation, the proper static waveshaper is chosen according to the V_D voltage. The V_D voltage is then actualized using

$$V_D[n] = \frac{-I_{a1} - I_{a2} - I_{s1} - I_{s2} + \frac{(V_{PS} - V_D[n-1])}{R_D}}{C_1 f_s} + V_D[n-1]. \tag{14}$$

3.1. Circuit with Loudspeaker Model

If the loudspeaker is connected to the power amplifier, the load impedance is no longer given explicitly, but it is determined by V_L , I_L relation implicitly given by equations (8). In order to exclude the loudspeaker impedance R_L , the first equation from system (13) is modified to

$$I_L[n] = \frac{N_1}{N_2} (I_{a1} - I_{a2}) \tag{15}$$

and then solution of the modified equations (13) with (15) is expressed as a function $I_L[n] = f_{IL}(V_{in1}[n], V_{in2}[n], V_D[n], V_L[n])$. Finally, the solution of the whole system using (9) is given by

$$-c_1 V_L[n] + I_{tmp} = f_{IL} \tag{16}$$

for the unknown variable $V_L[n]$ and state variables I_{tmp} and $V_D[n]$.

3.2. Circuit with Nonlinear Transformer Model

The simulation of a circuit with a nonlinear transformer model is based on (2), (3) and the nonlinear core model. The system is described using

$$\begin{aligned}
0 &= -H(B[n])l_{mag} - (c_1 V_L[n] + I_{tmp})N_2 + f_{IL}N_2 \\
0 &= B[n-1] - B[n] + \frac{V_L}{N_2 S f_s}
\end{aligned} \tag{17}$$

where term $f_{IL}N_2$ is recomputed current I_L to primary winding, $B[n-1]$ is the flux density in the previous sampling period and $H(B)$ is the core model derived from (4) or from (7) if the hysteresis is considered.

All the simulations from section 3 were implemented in Matlab environment using Mex files and C language. The Newton method was used for solving implicit nonlinear equations. The derivation of function or the Jacobian matrix were obtained using finite difference formula, the maximal number of iterations was 100 and the numerical error was chosen as 0.0001. The values for the transformer model were chosen experimentally: $M_s = 1.11 \times 10^6$, $3a = 8.56$, $\alpha = 8.82 \times 10^{-5}$, $c = 0.14$ and $k = 51.65$. However, they can be computed from a measured hysteresis loop data [11]. The parameters for the Frohlich core model were $c = 113.38$, $b = 0.71$ and the dimensions of the transformer were chosen as $S = 0.003 \text{ m}^2$ and $l_{mag} = 0.2 \text{ m}$.

The transformer-core cross-section S together with the number of windings determines the low cutoff frequency of the transformer. The simulation of the output part of the power amplifier was appended with a phase splitter, input pentode unit and feedback according to [1] and the results were compared to the measured Engl combo. The data was obtained from a measured voltage on a parallel loudspeaker output with connected soundcard. The frequency dependence of the first harmonic content of the voltage output signal, obtained using sweep sine signal, is shown in figure 3. The simulation with a constant load provides the worst results – there is no resonance around 45 Hz that appears in all of the other simulations and measured data. The simulation with hysteresis loop has similar behavior as a measured amplifier in the area of low frequencies due to the core losses. In the area of mid-frequencies, the simulation and measured data differ because of the other resonances and mechanical properties of the loudspeaker diaphragm, which are not considered in the simulation but can be included in the simulation by improving the loudspeaker model like in [13]. The nonlinear distortion was investigated as well. The output spectrum for an input sinewave signal with an amplitude of 2 V is shown in figures 4. The individual spectra are measured for the same frequency but they are shifted in the graph. All the algorithms excluding the variant with constant load give very similar results. The nonlinear distortion caused by transformer hysteresis manifests very slightly and only at frequencies below cca 150 Hz and it is very dependent on transformer parameters. The spectrogram of simulation using JA-model is shown in figure 5.

The table 2 shows the hypothetical computational complexity that was determined using measurement of time duration of simulation for input guitar riff signal with a length of 18 s and an amplitude of 30 V. The results showed that the algorithm is capable of working in real-time. Sound examples of all the algorithm variants and detailed graphs and other spectrograms are available on the web page www.utko.feec.vutbr.cz/~macak/DAFx11/.

Table 2: Normalized computational complexity.

Constant load	Loudspeaker	Frohlich	J-A model
0.04 %	0.05 %	0.12 %	0.27 %

5. CONCLUSIONS

The simulation of a push-pull tube amplifier was discussed in this paper. The impact of the loudspeaker and output transformer to

the amplifier properties was investigated. The loudspeaker was modeled using simplified electric circuit and Frohlich and Jiles-Atherton transformer models were used. The results showed that the output transformer had a minor impact on the simulation results. It only manifested at very low frequencies while the computational complexity was increased significantly. Both transformer models provided very similar simulation results and therefore, the Frohlich model is more efficient for real-time simulations.

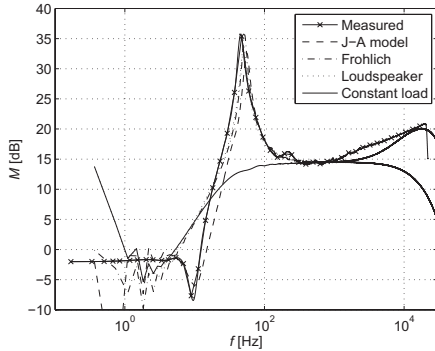


Figure 3: Frequency dependence of the first harmonic content of the output signal.

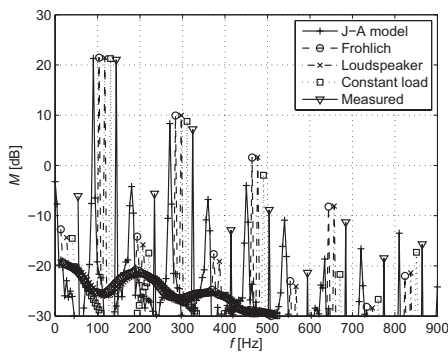


Figure 4: Output spectrum for input 90 Hz sinewave.

6. ACKNOWLEDGMENTS

This paper was supported by the Fund of the Council of Higher Education Institutions of the Czech Republic under project no. 2704/2011 and project no. FR-TI1/495 of the Ministry of Industry and Trade of the Czech Republic.

7. REFERENCES

[1] J. Macak and J. Schimmel, "Real-time guitar tube amplifier simulation using approximation of differential equations," in *Proceedings of the 13th International Conference on Digital Audio Effects DAFx10*, Graz, Austria, Sep. 6-10, 2010.

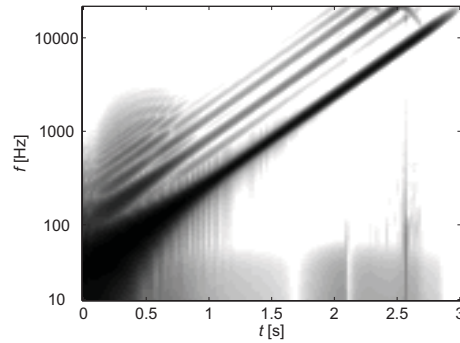


Figure 5: Spectrogram of output signal for simulation with JA model of transformer.

[2] D. Self et al, *Audio Engineering*, Elsevier, Burlington, MA, USA, 1st edition, 2009.

[3] U. Zölzer, *DAFX - Digital Audio Effects*, J. Wiley & Sons, Ltd, 1st edition, 2002.

[4] J. Pakarinen, M. Tikander, and M. Karjalainen, "Wave digital modeling the output chain of a vacuum-tube amplifier," in *Proc. Intl. Conf. on Acoustics, Speech, and Signal Proc.*, Como, Italy, Sept. 1-4, 2009, pp. 55-59.

[5] R. C. D. de Paiva, J. Pakarinen, V. Välimäki, and M. Tikander, "Real-time audio transformer emulation for virtual tube amplifiers," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, pp. 15, 2011.

[6] I. Cohen and T. Helie, "Real-time simulation of a guitar power amplifier," in *Proceedings of the 13th International Conference on Digital Audio Effects DAFx10*, Graz, Austria, Sep. 6-10, 2010.

[7] N. Koren, "Improved vacuum tube models for SPICE simulations," Available at http://www.normankoren.com/Audio/Tubemodspice_article.html, 2003.

[8] P. Kis, *Jiles-Atherton Model Implementation to Edge Finite Element Method*, Ph.D. thesis, Budapest University of Technology and Economics, 2010.

[9] S. E. Zocholl, A. Guzman, and D. Hou, "Transformer modeling as applied to differential protection," Tech. Rep., Schweitzer Engineering Laboratories, Inc. Pullman, Washington, 1999.

[10] D. C. Jiles and D. L. Atherton, "Ferromagnetic hysteresis," *IEEE Transactions on Magnetics*, no. 5, pp. 2183-2185, 1983.

[11] D. C. Jiles, J. B. Thoele, and M. K. Devine, "Numerical determination of hysteresis parameters for the modeling of magnetic properties using the theory of ferromagnetic hysteresis," *IEEE Transactions on Magnetics*, vol. 28, no. 1, pp. 27-35, 1992.

[12] R. Elliot, "Measuring Thiele / Small Loudspeaker Parameters," Available at <http://sound.westhost.com/tsp.htm>, 2007.

[13] J. Pakarinen, M. Tikander, and M. Karjalainen, "Wave digital modeling of the output chain of a vacuum-tube amplifier," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 1-4, 2009.

ANALYSIS AND TRANS-SYNTHESIS OF ACOUSTIC BOWED-STRING INSTRUMENT RECORDINGS – A CASE STUDY USING BACH CELLO SUITES

Yin-Lin Chen, Tien-Ming Wang, Wei-Hsiang Liao, Alvin W.Y. Su

SCREAM Lab., Department of CSIE,
National Cheng-Kung University
Tainan, Taiwan

p76981057@mail.ncku.edu.tw

ABSTRACT

In this paper, analysis and trans-synthesis of acoustic bowed string instrument recordings with new non-negative matrix factorization (NMF) procedure are presented. This work shows that it may require more than one template to represent a note according to time-varying behavior of timbre, especially played by bowed string instruments. The proposed method improves original NMF without the knowledge of tone models and the number of required templates in advance. Resultant NMF information is then converted into the synthesis parameters of the sinusoidal synthesis. Bach cello suites recorded by Fournier and Starker are used in the experiments. Analysis and trans-synthesis examples of the recordings are also provided.

Index Terms—trans-synthesis, non-negative matrix factorization, bowed string instrument

1. INTRODUCTION

In recent years, trans-synthesis is an interesting topic in musical processing [1-2]. Depending on either time-domain or frequency-domain properties of audio signal processing, the authors attempt to overcome the problems when the real audio recordings are analyzed, transformed, and re-synthesized. In particular, non-negative matrix factorization (NMF) is recently well-known to factorize spectrum into basis spectra and temporal activation in music signal analysis [3]. It is widely used for music transcription [4-5], pitch detection and onset detection. To improve original NMF, temporal smoothness [6], sparseness [7], and harmonicity/inharmonicity [8] have been considered as primary constraints. For accurate piano music transcription proposed in [9], trained note templates are obtained in advance.

[5] indicates that one needs to use enough number of required templates to give good results. In [8], 88 templates are used due to the pitch range of piano, and this takes lots of computation and memory space. In real case, however, the spectral behavior of one note played by an instrument is always time-varying, especially in the case of bowed string instruments. That means it is not reasonable to achieve the NMF task by using only one template per note. Fig. 1 shows the spectra of two different frame of the note B3 played by pianoUPM project [10]. It is interesting to note that spectral contours of two frames (20th frame and 300th frame in this case) are apparently in different shapes.

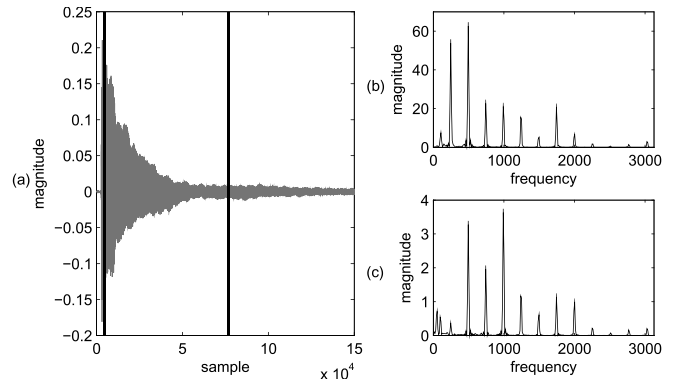


Figure 1: (a) The waveform of the note B3. Two bold lines indicated 20th and 300th frame. (b) The spectrum of 20th frame. (c) The spectrum of 300th frame.

Moreover, keyboard music is usually employed in the evaluations in previous works. How NMF performs for other types of instruments is seldom addressed. In this paper, an iterative procedure for deciding the appropriate number of templates is proposed for NMF. No trained tone model is required in advance. The proposed method is applied to acoustic cello recordings.

In order to reproduce the results of the proposed NMF procedure, appropriate synthesis technique is preferably applied. Spectral modeling synthesis (SMS) is proposed to divide music signal into deterministic part and stochastic part. This model was then extended by including transient modeling [11], called sinusoids plus transient and noise. In this paper, deterministic part of conventional SMS is applied to re-synthesize polyphonic musical signals analyzed by NMF. Sinusoidal synthesis [12] is employed here. Sound transformation examples are accomplished and provided in [13] as well.

This paper is organized as follows. In section 2, NMF is briefly reviewed and its modification is proposed. In section 3, experimental results of NMF analysis and music re-synthesis are given. Conclusion is given in section 4.

2. NMF-BASED MUSIC SIGNAL ANALYSIS

2.1. Brief review of NMF

In [3], given an $m \times n$ nonnegative matrix Y , NMF is used to factorize Y into an $m \times r$ nonnegative matrix W and an $r \times n$ nonnegative matrix X such that:

$$Y \approx \tilde{Y} = WX \quad (1)$$

A cost function such as KL divergence shown in equation 2 is designed as a measurement to evaluate how well the multiplication of W and X can approximate Y .

$$D_{KL}(Y|WX) = \left\| Y \otimes \log\left(\frac{Y}{WX}\right) - Y + WX \right\|_F \quad (2)$$

$\|\cdot\|_F$ is the Frobenius norm. \otimes is element-wise multiplication. By iteratively updating W and X , and the cost is minimized. For example, the update rules for (2) are shown as follows.

$$X_{rn} \leftarrow X_{rn} \frac{\sum_m W_{mr} Y_{mn} / (W \cdot X)_{mn}}{\sum_v W_{vr}} \quad (3)$$

$$W_{mr} \leftarrow W_{mr} \frac{\sum_n X_{rn} Y_{mn} / (W \cdot X)_{mn}}{\sum_k X_{rk}} \quad (4)$$

In music analysis, Y is used to represent signal spectrogram. For example, the column vector Y_j of Y is the spectrum of the j^{th} time frame. Frame size and total number of time frames are m and n , respectively. Hence, the column vector W_i of W represents the template of the i^{th} note contained in the signal, and the element X_{ij} of X indicates the intensity of the i^{th} note which appears in the j^{th} time frame.

For NMF, it is important to decide the required number of note templates, r , in advance, in order to give good factorization of Y . Two methods are usually used to decide r . One uses the number of different notes appearing in the signal [5]. The other sets r as 88 if piano music is analyzed [8]. However, the number of different notes is usually unknown and computation complexity is huge if $r = 88$.

Moreover, it is questionable if a template is enough for a piano tone. A sound clip containing 2 different notes obtained from [10] is analyzed. r is set as 2 and 88, respectively. NMF in [5] is used. The results are shown in Fig. 2. The left-hand-side figures represent X , and the right-hand-side represent the corresponding W . In Fig. 2(a), the 1st template contains both notes, and the 2nd template contains one of the notes. A note appears in both templates at the same time. It also shows that 2 templates are not enough to factorize the signal. In Fig. 2(b), only 4 templates give large enough intensity. The 1st note appears in the 33th and 45th templates, but with different spectrum envelopes. Similarly, the 2nd note appears in the 35th and 47th templates. It seems 88 templates are enough, but the computation time is huge.

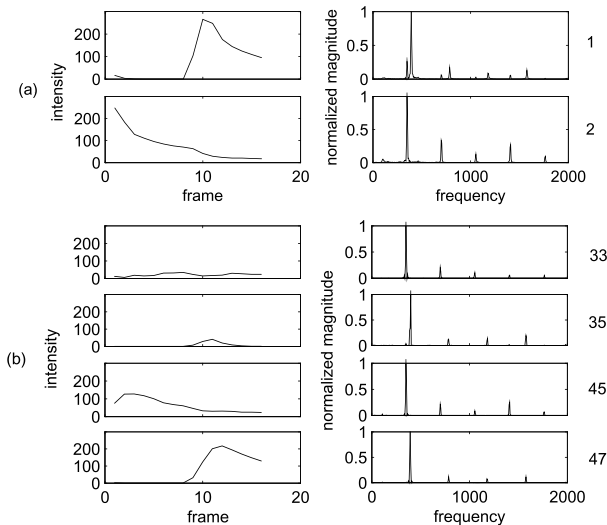


Figure 2: NMF for piano sound clip: (a) $r = 2$ (b) $r = 88$.

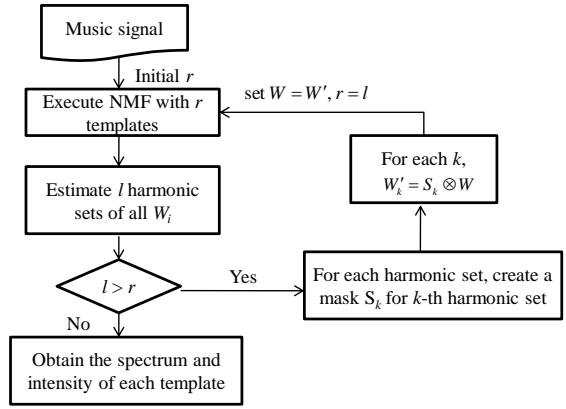


Figure 3: Flow chart of the proposed NMF procedure.

2.2. New NMF Procedure for Harmonic Music Signals

The proposed NMF depicted in Fig. 3 is proposed to solve the problem stated above without the knowledge of exact number of required templates. One first starts with a small r . The audio recording is then analyzed by means of NMF described in [5]. Two preliminary non-negative matrices, which stand for intensities and spectra of all templates, come out as result. Next, the spectrum of each template is evaluated to find if it contains more than one harmonic set by using the method in [14]. If so, a mask function is used to extract the k^{th} harmonic set. It can be represented as

$$W'_k = S_k \otimes W, \quad (5)$$

$$S_k = \sum_p (2\sigma) f(p \cdot \mu_k - \sigma, p \cdot \mu_k + \sigma; x),$$

where $f(x)$ is uniform distribution for the interval $[p \cdot \mu_k - \sigma, p \cdot \mu_k + \sigma]$, μ_k is the fundamental frequency corresponding to W'_k and p is partial index. σ is set as 3% of the fundamental frequency.

l represents the number of harmonic sets extracted from r templates. If $l \leq r$, the loop will be stopped and the eventual matrices are obtained. Otherwise, we set r as l , use the temporal matrices as initial conditions and NMF process is then executed again.

After the intensity and spectral information of all note templates are obtained, it can be found that FFT spectrum and \tilde{Y} in (1) are very close to each other (the result is shown in Section 3). Therefore, such NMF information can be used as parameters of the synthesis method stated below.

3. EXPERIMENTAL RESULTS

3.1. Results of the proposed NMF procedures.

Two music passages of Bach's cello suites No.1 (BWV1007) recorded by Starker [15] and Fournier [16] are analyzed. They are both polyphonic. There are 4 different pitches in the first 16 notes, shown in Fig.4. Cost function in (2) is implemented. Frame size is 8192 and hop size is 2048. Initially, the number of template, r , is set as 4. Templates are initialized with random numbers. 100 iterations are performed for NMF update rules. The outer loop in Fig. 3 runs only twice to reach the final results in both cases.

After obtaining the NMF result, W and X are used to obtain synthesis parameters, described in Section 3.2 due to the similarity between Y and \tilde{Y} . The 95th frame of Y and \tilde{Y} are shown in Fig. 5. It shows that the envelopes of two representations are close.



Figure 4: Score of first 16 notes.

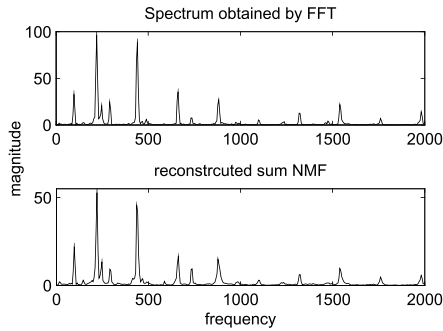


Figure 5: FFT spectrum and \tilde{Y} of the 95th frame obtained from the Fournier recording.

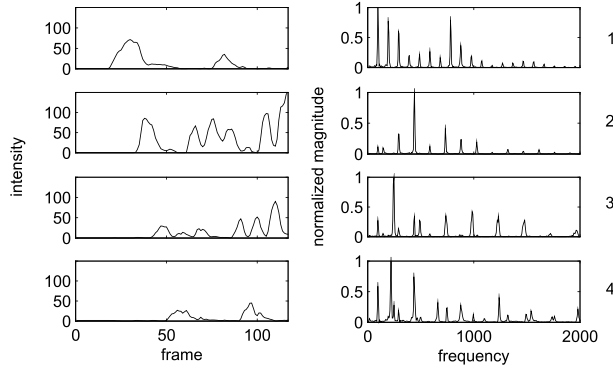


Figure 6: NMF result when the 1st outer loop factorization is finished ($r=4$). [Fournier]

Fig.6 shows the results of conventional NMF procedure of the Fournier recording with 4 initial templates ($r = 4$). In this case, more than one harmonic set exists in 3rd and 4th template such that these templates cannot completely represent all notes of the recording. One of the intuitive ways is to extend template dimension and makes these additional harmonic sets be located in new templates.

The final results of the proposed NMF procedure of the Starker and Fournier recordings are shown in Fig. 7 and 8 respectively. In the figures, each column vector W_i is normalized such that the intensity information is solely represented by the corresponding row vector X_i of X as

$$W_i = \frac{W_i}{\max(W_i)}, \quad X_i = X_i \cdot \max(W_i). \quad (6)$$

The note information is stated as follows. In Fig. 7, pitches of the first three templates are G2, the 4th to the 6th ones are D3, the 7th and the 8th ones are B3, the 9th one is A3, and the last template with no obvious f_0 is regarded as noise. In Fig. 8, pitches of the first three templates are G2, the 4th and the 5th ones are D3, the 6th and the 7th ones are B3, the 8th one is A3, and the last template with no obvious f_0 is also regarded as noise.

By taking the first three templates of Fig. 7 as examples, both of their spectral envelopes and the corresponding intensity functions are quite different. The 1st template can be regarded as the attack template of G2 notes because it has the largest intensity of three and its onset points appear early. Moreover, the duration of the 2nd template is longer and the onsets of the 3rd template appear between those of the 1st and the 2nd ones. It is reasonable to regard them as a sustain template and a decay template respectively. It is interesting to note that the number of templates is solely related to the waveform behavior of the note. It usually depends on the physical architecture of the instrument and the gestures of the musician while he/she playing that note. Three templates are needed to represent G2 notes in this case. The required numbers of templates of D3 notes in the two recordings are different (2 and 3 for the Fournier and Starker recording, respectively).

Our experiments described above show that to use one template for a note may not be enough to sufficiently model a note, especially when the information is to be used in the re-synthesis process. It is interesting to see that when r is set as 10 without using the proposed procedure, one can't successfully factorize the signal from the Starker recording. Due to the harmonic mask $S_{i,k}$, the proposed NMF procedure outperforms in this case. Comparing to Fig. 7, the 6th and the 9th templates contain 2 notes in Fig. 9. This shows that the proposed method is advantageous even when the number of templates is enough.

3.2. Resynthesis

Eventually, W and X are converted to parameters of sinusoids. The number of partials is set as 50. The original and the synthetic signals are compared in Fig.10. The two spectral envelopes are close. Sound transformation such as timbre modification can be easily accomplished by replacing NMF templates of the corresponding notes. Sound examples are provided in [13].

4. CONCLUSION AND DISCUSSION

Analysis and trans-synthesis of acoustic cello recordings made by Fournier and Starker with modified NMF procedure is presented. It is not required to have pre-trained tone model and to know the necessary number of templates in advance for the modified NMF to give good results. It is also found that more than one template can be used to preferably represent a note according to its different sounding states. Spectrum and intensity information of NMF is then converted into the synthesis parameters of the sinusoids. Trans-synthesis sound examples of Bach cello suites can be heard in [13].

Comparing to other state-of-art methods, this paper puts emphasis on the applications of sound reproduction rather than music transcription. That means spectral behavior of the timbre is more significant than the statistical results like F-measure or mean overlap ratio. Without applying any temporal or spectral constraints on NMF update rules, the proposed method models each note by extending template dimension such that these templates can be in charge of different states of one note. According to the aspect, the adequate number of templates will be unpredictable if one music note is played by various kinds of instruments with different gestural representations. It therefore takes amount of computation by means of the iteratively procedure of harmonic verification and multiple NMF updates.

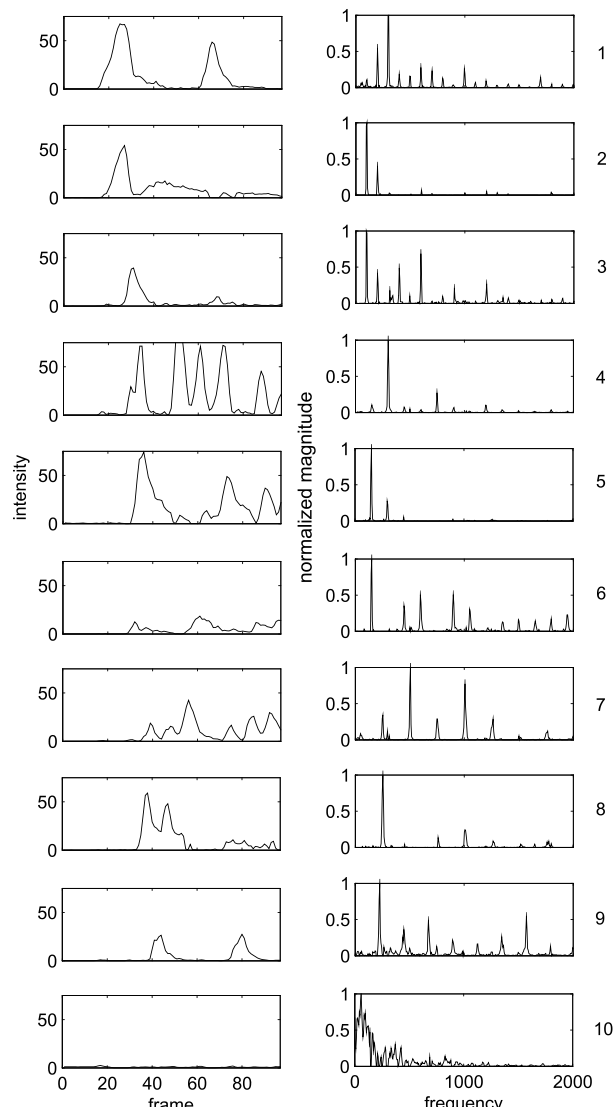


Figure 7: the result of the proposed NMF procedure. [Starker]

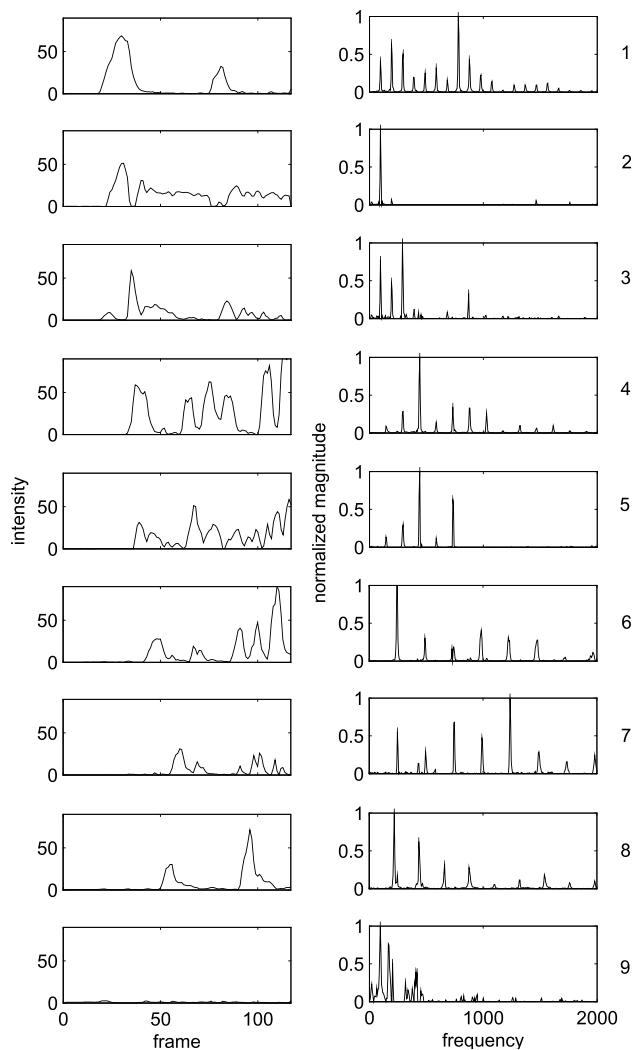


Figure 8: the result of the proposed NMF procedure. [Fournier]

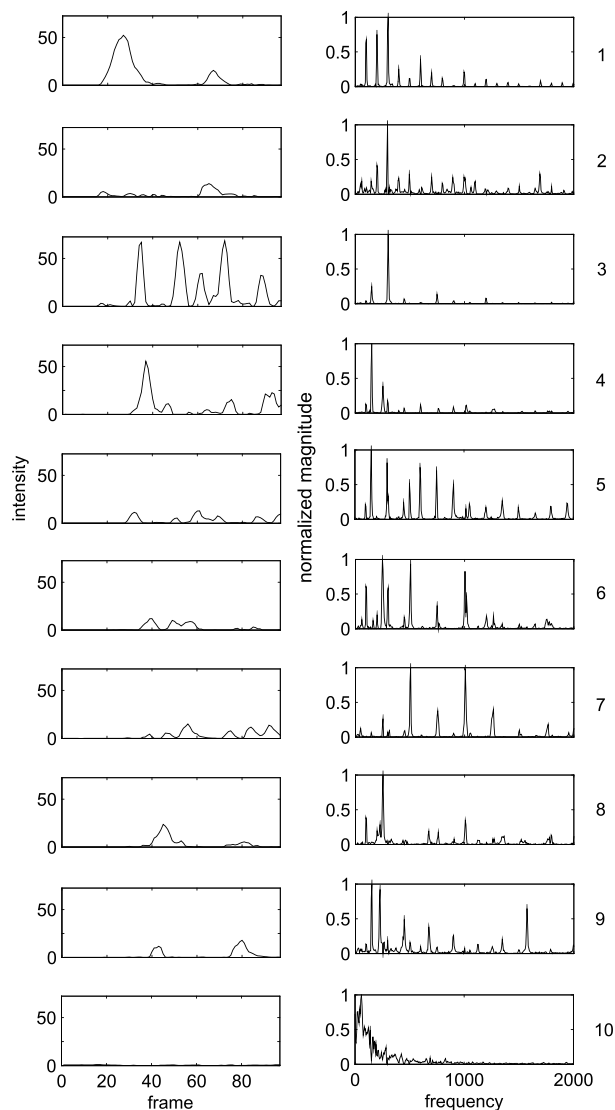


Figure 9: the result of convention NMF procedure ($r=10$). [Starker]

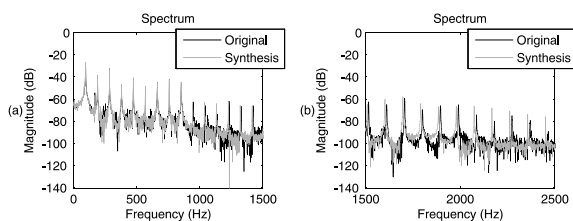


Figure 10: FFTs of (a) lower partials and (b) higher partials of 32th original and synthetic frame of the Fournier recording.

The modeling of time-varying templates is one of possible improvements. The refinement of temporal prior may suit for modeling the sustain and decay parts of the note. The attacks or other transient parts are however disfavored. Therefore, the templates are considerable to ‘morph’ as time goes by when the notes are activated. A time-varying multiplicative gradient approach with adaptive templates may be investigated on in the future.

5. REFERENCES

- [1] W.-C. Chang, Y.-S. Siao, and W.-Y. Su, “Analysis and transynthesis of solo Erhu recordings using additive/subtractive synthesis,” in *120th Audio Engineering Society (AES) Convention*, Paris, France, 2006.
- [2] T.-M. Wang, W.-C. Chang, K.-T. Lin *et al.*, “Trans-Synthesis System for Polyphonic Musical Recordings of Bowed-String Instruments,” in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, 2009.
- [3] D. Lee, and H. Seung, “Algorithms for Non-negative Matrix Factorization,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 13, pp. 556-562, 2001.
- [4] N. Bertin, R. Badeau, and E. Vincent, “Enforcing harmonicity and smoothness in Bayesian non-negative matrix factorization applied to polyphonic music transcription,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 3, pp. 538-549, 2010.
- [5] P. Smaragdis, and J. Brown, “Non-negative matrix factorization for polyphonic music transcription,” in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 2003, pp. 177-180.
- [6] T. Virtanen, “Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 3, pp. 1066-1074, 2007.
- [7] P. O. Hoyer, “Non-negative matrix factorization with sparseness constraints,” *The Journal of Machine Learning Research*, vol. 5, pp. 1457-1469, 2004.
- [8] E. Vincent, N. Berlin, and R. Badeau, “Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription,” in *Proc. of International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, Nevada, USA, 2008, pp. 109-112.
- [9] A. Dessein, A. Cont, and G. Lemaitre, “Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence,” in *proc. 11th International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010.
- [10] L. Ortiz-Berenguer, E. Blanco-Martin, A. Alvarez-Fernandez *et al.*, “A Piano Sound Database for Testing Automatic Transcription Methods,” in *125th AES Convention*, San Francisco, 2008.
- [11] T. Verma, and T. Meng, “An Analysis/Synthesis Tool for Transient Signals that Allows a Flexible Sines+ Transients+ Noise Model for Audio,” in *proc. IEEE Conf. on Acoustics, Speech and Signal Processing*, 1998.
- [12] R. J. McAulay, and T. F. Quatieri, “Speech Analysis Synthesis Based on a Sinusoidal Representation,” *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 34, no. 4, pp. 744-754, Aug, 1986.
- [13] T.-M. Wang, “Experimental Results,” <http://www.scream.csie.ncku.edu.tw/index.php/research/asp/dafx2011exp>.
- [14] A. P. Klapuri, “Multiple fundamental frequency estimation based on harmonicity and spectral smoothness,” *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, pp. 804-816, 2003.
- [15] J. Starker, “Bach: Six Suites; Sonatas in G & D,” Mercury Records, CD 1, Track 1, 1991.
- [16] P. Fournier, “Bach: 6 Suiten für Violoncello solo,” Archiv Records, CD 1, Track 1, 1997.

DHM AND FDTD BASED HARDWARE SOUND FIELD SIMULATION ACCELERATION

Yasushi Inoguchi, Tan Yiyu, Yukinori Sato

Center for Information Science, Japan
Advanced Institute of Science & Technology
Ishikawa, Japan

inoguchi@jaist.ac.jp

yiyu-t@jaist.ac.jp

yukinori@jaist.ac.jp

Yukio Iwaya, Hiroshi Matsuoka,

Research Institute of Electrical
Communication, Tohoku University
Sendai, Japan

iwaya@riec.tohoku.ac.jp

matsuoka@riec.tohoku.ac.jp

Makoto Otani

Faculty of Engineering, Shinshu University

Nagano, Japan

otani@cs.shinshu-u.ac.jp

Takao Tsuchiya

Faculty of Science and Engineering, Doshisha
University

Kyoto, Japan

ttsuchiya@mail.doshisha.ac.jp

ABSTRACT

Sound field simulation is widely used for acoustic design; however, this simulation needs many computational resources. On the other hand, FPGA becomes major for acceleration. To take advantage of hardware acceleration by FPGA, hardware oriented algorithm which consumes small number of gates and memory is necessary. This paper addresses hardware acceleration of sound field simulation using FPGA. Improved Digital Huygens Model (DHM) for hardware is implemented and speed up ratio is examined. For 2D simulation, the implemented accelerator is 1,170 times faster than software simulation. For 3D simulation, it is shown that FDTD based method is suitable for hardware implementation and required hardware resource are estimated.

1. INTRODUCTION

Simulation of sound field is useful for acoustic design. Recently, performance of computer system is much improved and many researches to simulate sound field numerically are proposed. Wave based numerical analyses such as finite element^[1], finite boundary element^[2], finite difference time domain (FDTD) method^[3-4] are major ways for simulation. Especially FDTD method is one of the best way to analyze sound field because this method calculates sound pressure at any place in the simulation field directly and it gives good accuracy. Recently, some of physical equivalent methods are also proposed^[5-7] to investigate acoustical behavior. These methods are derived from the wave equations of sound propagation, and Digital Huygen's Model (DHM) is one of these methods.

Concept of DHM have been initially proposed by S.A. Van Duyne and J.O. Smith^[6-7] and expanded by T. Tsuchiya et al.^[8-9] DHM is modification of FDTD for hardware implementation. It assumes that sound pressure pulse propagates between acoustic tubes to neighboring nodes. When a pulse is given to a node, scatters are sent to four adjacent nodes. Each node takes scatters from adjacent nodes and calculates its pressure. Iterating this procedure independently at each node, we can simulate sound pressures at all nodes. However, DHM also requires huge computational resources and it is difficult to simulate in real time.

On the other hand, recently Field Programmable Gate Array (FPGA) becomes popular to make a special purpose machines. Internal circuits of FPGA chip can be programmed by users, and hundreds percent of logical resources on a chip can be used for users' purpose. Several implementations using FPGA have been proposed^[10-12,15-17]. Chuan et al.^[10] reported 1.5 to 4 times acceleration and Motuk et al.^[11] reported 10 to 35 times acceleration.

Accelerating by GPGPU is another solution and several researches are reported^[13-14] but we don't treat GPGPU in this paper due to the limitation of pages.

In this research we introduce new implementation of real-time sound field simulation system using FPGAs. Hardware oriented two-dimensional DHM implementation and three-dimensional FDTD implementation are shown. Simulation by FPGA gives very good performance because all calculation is executed in a chip and all computing elements perform calculation simultaneously. Techniques to reduce gate consumption are also developed. Our system for 2D simulation is over than a thousand times faster than simulation by software.

Following is structure of this paper. Section 2 shows hardware implementation of a two-dimensional DHM algorithm, including algorithm optimization to reduce operations, and evaluation of calculation precision and speed. In section 3, three-dimensional FDTD implementation is discussed. Techniques to reduce circuit are introduced and number of chips to simulate a small chamber is shown. Chapter 4 is conclusion of this paper.

2. 2D DHM FOR HARDWARE IMPLEMENTATION

2.1. Basic of 2D DHM

The DHM is sound field simulation method for hardware. In the DHM, sound field is mapped on a grid, and each node of grid keeps its pressure. Figure 1 shows idea of the two-dimensional DHM. Figure 1 (a) shows that each node adjoins four neighbors by acoustic tubes which length is Δl . Assume that an incident is injected to a node. Then, sound pressure at the node is scattered through acoustic tubes as shown in figure 1(b). To make discussion easy, four adjacent tubes called as E, W, N, S, shown in figure 1 (c).

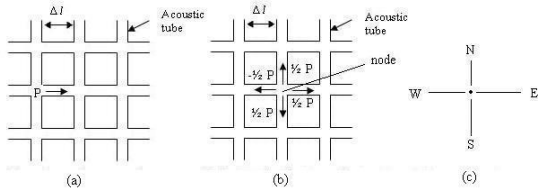


Figure 1: Concept of 2D Digital Huygen's Model. (a) shows mapping on a grid and unit length of an acoustic tube, (b) shows scattering at a node, and (c) shows direction definition.

A pressure at node (i,j) at time step n is denoted as $\phi^n(i,j)$, the incident will be transmitted to four neighboring nodes. Scattered pulse $S_m^n(i,j)$ from line $m \in \{E, W, N, S\}$ at the time step n at location (i,j) and sound pressure $\phi^n(i,j)$ are calculated by the following equations.

$$\begin{bmatrix} S_E^n(i,j) \\ S_W^n(i,j) \\ S_N^n(i,j) \\ S_S^n(i,j) \end{bmatrix} = \phi^n(i,j) - \begin{bmatrix} P_E^n(i,j) \\ P_W^n(i,j) \\ P_N^n(i,j) \\ P_S^n(i,j) \end{bmatrix} \quad (1)$$

$$\phi^n(i,j) = \frac{1}{2} (P_E^n(i,j) + P_W^n(i,j) + P_N^n(i,j) + P_S^n(i,j)) \quad (2)$$

where $P_m^n(i,j)$ are incident pulses. A scatter on each line becomes an incidence to the neighboring nodes. Thus, incidents at the next time step $n+1$ are calculated by following equation.

$$\begin{bmatrix} P_E^{n+1}(i,j) \\ P_W^{n+1}(i,j) \\ P_N^{n+1}(i,j) \\ P_S^{n+1}(i,j) \end{bmatrix} = \begin{bmatrix} S_E^n(i-1,j) \\ S_W^n(i+1,j) \\ S_N^n(i,j-1) \\ S_S^n(i,j+1) \end{bmatrix} \quad (3)$$

Through equations (1) to (3), scatters and pressure of all nodes can be calculated at every time step whose interval is $\Delta l/c$ where c is sound speed.

2.2. Hardware Resources to Implement 2D DHM

From point of view of implementation of this algorithm on a hardware circuit, you can see that only calculation of equations (1) and (2) are necessary. Equation (3) is implemented by variable renaming or wire connection and no operation is required. Four subtractors are needed to calculate equation (1), three adders and one right shifter are needed to calculate equation (2), and the right shifter to calculate $1/2$ in equation (2).

Each node must keep four incidences, four scatters and one sound pressure. However, either incidences or scatters are necessary because incidences are calculated by scatters of the neighboring nodes. Thus five temporary values must be kept at each node.

2.3. Hardware Oriented 2D DHM

Equations (1) and (2) are given by the original DHM algorithm. However, we can reduce number of operations of this algorithm. We eliminate incident $P_m^n(i,j)$ by inserting equations (1) and (3) into equation (2) and obtain a following equation.

$$\begin{aligned} \phi^n(i,j) = \frac{1}{2} & \left(\phi^{n-1}(i-1,j) + \phi^{n-1}(i+1,j) \right. \\ & + \phi^{n-1}(i,j-1) + \phi^{n-1}(i,j+1) \\ & \left. + \sum_m S_m^{n-2}(i,j) \right) \end{aligned} \quad (4)$$



Figure 2: Photograph of the hardware accelerator using FPGAs.

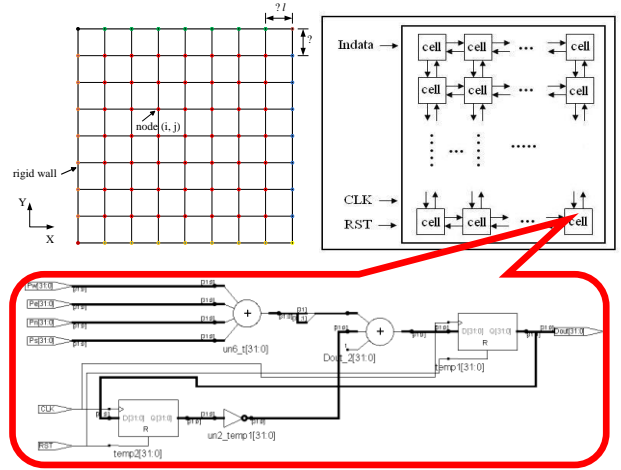


Figure 3: Structure of hardware accelerator. Sound pressure at each acoustic node is calculated by a computing cell synthesized on a FPGA chip.

Since scattering matrix is symmetrical, we can assume:

$$\sum_m S_m^n(i,j) = \sum_m P_m^n(i,j) \quad (5)$$

Using this relation, equation (4) can be written as

$$\begin{aligned} \phi^n(i,j) = \frac{1}{2} & (\phi^{n-1}(i-1,j) + \phi^{n-1}(i+1,j) \\ & + \phi^{n-1}(i,j-1) + \phi^{n-1}(i,j+1)) - \phi^{n-2}(i,j) \end{aligned} \quad (6)$$

Equation (6) shows that we need only three adders, one subtractor and one right-shifter. From point of view of memory, each node must keep two sound pressure (32bit) at each time steps.

2.4. Hardware Implementation of 2D DHM

Figure 2 shows the photograph of the hardware accelerator based on the 2D DHM. The system is SPP3000 provided by Tokyo Electron Device Ltd. and consists of several FPGAs and one CPU board to control system and program FPGAs. Each FPGA chip keeps a lot of calculation cells based on equation (6). In the figure, two inserted boards on the left side are FPGA boards and each board has two Xilinx XC5LVX330T-FF1738 FPGA chips and 512Mbit SRAM. This FPGA chip contains 51,840 Configu-

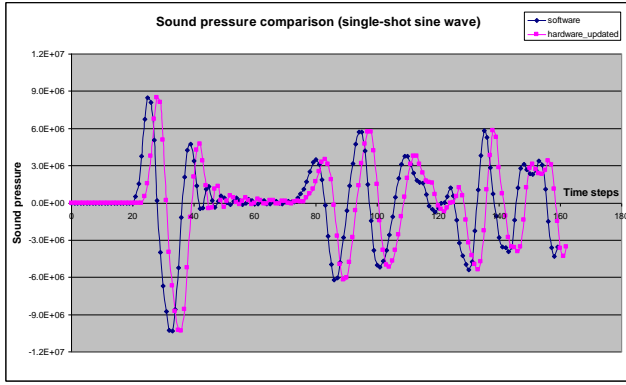


Figure 4: Sample of simulation result. Hardware simulated result is three-cycles delayed but both result are completely equal if software result puts off three cycles.

rable Logic Blocks (CLBs) and 11,664Kbit Block RAM. A board on the right is a CPU board which has single 1.4GHz Intel Pentium-M processor with 504MB memory. FPGA boards and host PC are connected by the compact PCI bus. Calculation for simulation is operated on the FPGA boards and host PC is used only for program and input/output data.

Figure 3 shows outline of our implementation. As shown in the left half of the figure, a sound field is mapped to a grid and each node on the grid calculates equation (6). Each node has actually implemented as calculation cell shown in right half of the figure. Calculation cell has a set of adders and registers. As shown in the schematic, one 4-inputs adder, one 3-inputs adder, and two 32-bit registers are used. A 4-inputs adder is actually implemented as three 2-inputs adders.

2.5. Precision Evaluation

To examine precision of the hardware simulation, we compared simulation result of hardware with software. Improved 2D DHM is programmed by C++ and executed by a software using an AMD Phenom 9500 1.8GHz Quad-core processor with 4GB memory. The simulation space is 32x32 and sampling rate is 16. Same algorithm is implemented on a FPGA system.

Incident point of signal is node (0,0) and simulation result is taken from node (6,15). In this evaluation, a single-shot sine wave and Gaussian wave are used and magnitude of them is 10^8 Pa. Both results of hardware and result of software are completely same. Figure 4 shows a sample of simulation result. Hardware result is delayed three clocks due to circuit initialization.

2.6. Evaluation of Simulation Speed

Table 1 shows comparison of simulation speed while 20,000 time steps. We examined three software simulation and a hardware simulation. In the table, “original DHM” shows time of simulation of DHM by software using equations (1) and (2). “Updated DHM” uses new DHM algorithm based on the equation (6) and its simulation time is almost half of original one. FDTD indicates the time of simulation using FDTD method. Hardware solution shows time of simulation based on equation (6) using FPGA. Hardware simulate system consist of two stages. One is calculation stage and another one is input/output (I/O) stage. This shows that calculation stage takes only 0.2ms and this speed is

Table 1: Comparison of simulation time for 2D simulation.

Nodes	Time steps	software solution			hardware solution
		original DHM	updated DHM	FDTD	updated DHM
10 x 10	20000	31ms	30ms	109ms	0.2ms (50ms)
25 x 25	20000	234ms	124ms	266ms	0.2ms (50ms)
32 x 32	20000	328ms	234ms	406ms	0.2ms (50ms)

almost 1,170 times faster than software simulation using same method. I/O stage takes much longer and time for simulation with I/O takes almost 50ms. Since I/O is only needed at initialization and finalization of a simulation, time for I/O can be ignored if the simulated time steps are much longer.

3. 3D FDTD FOR HARDWARE IMPLEMENTATION

3.1. 3D Hardware Acceleration of Sound Field Simulation

Although hardware implemented 2D-DHM is over a thousand times faster than software implementation, this scheme doesn't efficient at 3D. Division by 2 in equation (2) can be implemented as a 1-bit right-shifter and it is very simple circuit, however, division by three is needed for 3D and there is no simple way to implement this circuit. Thus, we use different scheme based on FDTD for 3D to eliminate division by three.

3.2. Basic Scheme of FDTD

Finite-difference time-domain (FDTD)^[3-4] method is another basic way to simulate sound field. In this section we introduce efficient architecture to implement FDTD on hardware system. Governing equations for linear sound propagation are

$$\frac{\partial \phi}{\partial t} = -\rho c^2 \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) \quad (7)$$

$$\frac{\partial u_x}{\partial t} = -\frac{1}{\rho} \frac{\partial p}{\partial x}, \quad \frac{\partial u_y}{\partial t} = -\frac{1}{\rho} \frac{\partial p}{\partial y}, \quad \frac{\partial u_z}{\partial t} = -\frac{1}{\rho} \frac{\partial p}{\partial z} \quad (8)$$

where ϕ is sound pressure, t is time, ρ is medium density and c is sound speed. Since discretized FDTD is well known, we introduce a hardware oriented FDTD as follows. Sound pressure ϕ at (i,j,k) at time step n is given by:

$$\begin{aligned} \phi^n(i,j,k) &= \phi^{n-1}(i,j,k) \\ &\quad - \chi^2 \left\{ \bar{u}_x^{n-\frac{1}{2}} \left(i + \frac{1}{2}, j, k \right) - \bar{u}_x^{n-\frac{1}{2}} \left(i - \frac{1}{2}, j, k \right) \right. \\ &\quad + \bar{u}_y^{n-\frac{1}{2}} \left(i, j + \frac{1}{2}, k \right) - \bar{u}_y^{n-\frac{1}{2}} \left(i, j - \frac{1}{2}, k \right) \\ &\quad \left. + \bar{u}_z^{n-\frac{1}{2}} \left(i, j, k + \frac{1}{2} \right) - \bar{u}_z^{n-\frac{1}{2}} \left(i, j, k - \frac{1}{2} \right) \right\} \end{aligned} \quad (9)$$

$$\bar{u} = \frac{\rho c}{\chi} u \quad (10)$$

$$\Delta x = \Delta y = \Delta z = \Delta l, \quad \chi = \frac{c \Delta t}{\Delta l} = \frac{c}{\Delta l f} \leq \frac{1}{\sqrt{3}} \quad (11)$$

where Δt is the unit time, $\Delta x, \Delta y, \Delta z$ are the unit length.

$$\begin{aligned} \bar{u}_x^{n+\frac{1}{2}} \left(i + \frac{1}{2}, j, k \right) &= \bar{u}_x^{n-\frac{1}{2}} \left(i + \frac{1}{2}, j, k \right) \\ &\quad - \{ \phi^n(i+1, j, k) - \phi^n(i, j, k) \} \end{aligned} \quad (12)$$

$\bar{u}_y^{n+\frac{1}{2}}$ and $\bar{u}_z^{n+\frac{1}{2}}$ are also calculated as well as equation (12).

3.3. Assumption for Hardware Implementation

When we assume $\chi = \frac{1}{\sqrt{3}}$, this is usually the most general assumption, solving equation (9) requires division by 3. However, from point of view of hardware implementation, a divider takes many clock cycles and large amount of gates, and spoils efficiency of acceleration.

Assume χ is 1/2 instead of $\frac{1}{\sqrt{3}}$, and this assumption satisfies condition (11). The coefficient in equation (9) is changed to 1/4. On hardware system, a divider by four is easily implemented by a 2-bit right-shifter.

3.4. Estimation of Required Hardware resources

We estimate exact value of χ , Δl , and Δt . Sound speed c is 340m/s. In this implementation, maximum frequency to be simulated is 3KHz and we use 5 times over sampling to avoid error. Thus, Δt is $\frac{1}{3\text{KHz} \times 5} = 66.7\mu\text{s}$. In case of $\chi = \frac{1}{\sqrt{3}}$, Δl is 39.3mm and Δl is 45.3mm when $\chi = \frac{1}{2}$.

Now, we estimate the number of sound nodes to simulate a chamber which size is $2m \times 2m \times 4m$. 264k nodes are required when $\Delta l = 39.3\text{mm}$, and 172k nodes are required when $\Delta l = 45.3\text{mm}$, respectively. Our proposed assumption also reduces the number of sound nodes.

From viewpoint of memory consumption, each node keeps four variables: $p^n(i, j, k)$, $\bar{u}_x^{n+\frac{1}{2}}$, $\bar{u}_y^{n+\frac{1}{2}}$, and $\bar{u}_z^{n+\frac{1}{2}}$. Since each variable takes 32 bits, totally 256bits is needed for a node. Since XC5VLX330T has 11,664Kbit block RAM, one FPGA chip can accommodate 45.6K nodes in a chip. To simulate a chamber described above, 6 FPGA chips are needed if $\chi = \frac{1}{\sqrt{3}}$, and 4 chips are needed if $\chi = \frac{1}{2}$.

4. CONCLUSIONS

In this paper we proposed an implementation of hardware accelerated sound field simulator based on Digital Huygen's Model (DHM) for two-dimensional sound spaces. Improved DHM scheme requires very simple circuit and needs only four integer adders and a shifter. The proposed algorithm was implemented on a FPGA system and it was shown that hardware accelerated system is one thousand times faster than software simulation excluding I/O.

To simulate 3D, FDTD based implementation is better than DHM because FDTD based scheme can avoid synthesis of divider. From point of view of memory, it was shown that 45.6K nodes are accommodated in one FPGA chip.

To implement 3D simulator based on FDTD is our future work.

5. REFERENCES

- [1] J. N. Reddy, An Introduction to the Finite Element Method, 3rd ed., McGraw-Hill, USA, 2006.
- [2] P. K. Banerjee, and R. Butterfield, The Boundary Element Methods in Engineering Science, McGraw-Hill, USA, 1994.
- [3] D. Botteldooren, "Acoustical finite-difference time-domain simulation in a quasi-cartesian grid", Journal of the Acoustical Society of America, vol. 95, no. 5, pp. 2313–2319, 1994.
- [4] D. Botteldooren, "Finite-difference time-domain simulation of lowfrequency room acoustic problems", Journal of the

Acoustical Society of America, vol. 98, no. 6, pp. 3302–3308, 1995.

- [5] G. R. Campos and D. M. Howard, "On the computational efficiency of different waveguide mesh topologies for room acoustic simulation", IEEE Transactions on Speech and Audio Processing, vol. 13, no. 5 pp. 1063–1072, 2005.
- [6] S. A. Van Duyne, and J. O. Smith III, "The 2D digital waveguide mesh", In Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 1993, pp. 177–180.
- [7] S. A. van Duyne, J. R. Pierce, and J. O. Smith III. "Traveling Wave Implementation of a Lossless Mode-Coupling Filter and the Wave Digital Hammer". In Proc. Int. Computer Music Conf. (ICMC'94), pp. 411–418, 1994
- [8] Y. Kagawa, T. Tsuchiya, B. Fujii, and K. Fujika, "Discrete Huygens' model approach to sound wave propagation", Journal of Sound and Vibration, vol. 218, no. 3, pp. 419–444, 1998.
- [9] T. Tsuchiya, "Numerical simulation of sound wave propagation with sound absorption using digital Huygens' model", Japanese Journal of Applied Physics, vol. 46, no. 7B, pp.4809–4812, 2007.
- [10] Chuan He, Wei Zhao, and Mi Lu, "Time Domain Numerical Simulation for Transient Waves on Reconfigurable Coprocessor Platform", Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)
- [11] E. Motuk, R. Woods, S. Bilbao and J. McAllister. "Design Methodology for Real-Time FPGA-Based Sound Synthesis", IEEE Trans. on Signal Processing, Vol. 15, No. 12, pp. 5833–5845, 2007
- [12] Campos, Guilherme; Barros, Sara, "The Meshotron: A Network of Specialized Hardware Units for 3-D Digital Waveguide Mesh Acoustic Model Parallelization", In Proc of 128th AES Convention, pp. 8054, 2010.
- [13] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low- and mid-frequency room acoustics", In Proc. Int. Conf. on Digital Audio Effects, 2010.
- [14] Craig J. Webb, and S. Bilbao, "Virtual Room Acoustics : A Comparison of techniques for computing 3D-FDTD schemes using CUDA", In Proc of the 130th Convention, 7 pages, 2011.
- [15] T. Tsuchiya, E. Sugawara and Y. Inoguchi, "On the numerical simulation of sound wave propagation using field programmable gate array device", Journal of the Acoustical Society of America, vol. 120, no. 5, pp. 3218–3218, 2006.
- [16] T. Yiyu, Y. Inoguchi, E. Sugawara, Y. i Sato, M. Ohya, Y. Iwaya, H. Matsuoka and T. Tsuchiya, "A FPGA Implementation of the Two-Dimensional Digital Huygens' Model", International Conference on Field-Programmable Technology (FPT'10), pp.304–307, Beijing, China, Dec. 8–10, 2010
- [17] T. Yiyu, Y. Sato, E. Sugawara, Y. Inoguchi, M. Ohya, Y. Iwaya, H. Matsuoka, T. Tsuchiy, "A real-time sound field renderer based on digital Huygens' model", submitted to Journal of Sound and Vibration.

6. ACKNOWLEDGEMENT

This research is supported by National Institute of Information and Communications Technology.

SINUSOID EXTRACTION AND SALIENCE FUNCTION DESIGN FOR PREDOMINANT MELODY ESTIMATION

Justin Salamon, Emilia Gómez and Jordi Bonada, *

Music Technology Group

Universitat Pompeu Fabra, Barcelona, Spain

{justin.salamon, emilia.gomez, jordi.bonada}@upf.edu

ABSTRACT

In this paper we evaluate some of the alternative methods commonly applied in the first stages of the signal processing chain of automatic melody extraction systems. Namely, the first two stages are studied – the extraction of sinusoidal components and the computation of a time-pitch salience function, with the goal of determining the benefits and caveats of each approach under the specific context of predominant melody estimation. The approaches are evaluated on a data-set of polyphonic music containing several musical genres with different singing/playing styles, using metrics specifically designed for measuring the usefulness of each step for melody extraction. The results suggest that equal loudness filtering and frequency/amplitude correction methods provide significant improvements, whilst using a multi-resolution spectral transform results in only a marginal improvement compared to the standard STFT. The effect of key parameters in the computation of the salience function is also studied and discussed.

1. INTRODUCTION

To date, various different methods and systems for automatic melody extraction from polyphonic music have been proposed, as evident by the many submissions to the MIREX automatic melody extraction evaluation campaign¹. In [1], a basic processing structure underlying melody extraction systems was described comprising three main steps – multi-pitch extraction, melody identification and post-processing. Whilst alternative designs have been proposed [2], it is still the predominant architecture in most current systems [3, 4, 5, 6]. In this paper we focus on the first stage of this architecture, i.e. the multi-pitch extraction. In most cases this stage can be broken down into two main steps – the extraction of sinusoidal components, and the use of these components to compute a representation of pitch salience over time, commonly known as a *Salience Function*. The salience function is then used by each system to determine the pitch of the main melody in different ways.

Whilst this overall architecture is common to most systems, they use quite different approaches to extract the sinusoidal components and then compute the salience function. For extracting sinusoidal components, some systems use the standard Short-Time Fourier Transform (STFT), whilst others use a multi-resolution transform in an attempt to overcome the time-frequency resolution trade-off inherent to the FFT [7, 8, 9]. Some systems apply filters to the audio signal in attempt to enhance the spectrum of the

melody before performing spectral analysis, such as bandpass [7] or equal loudness filtering [6]. Others apply spectral whitening to make the analysis robust against changes in timbre [3]. Finally, given the spectrum, different approaches exist for estimating the peak frequency and amplitude of each spectral component.

Once the spectral components are extracted, different methods have been proposed for computing the time-frequency salience function. Of these, perhaps the most common type is based on harmonic summation [3, 4, 5, 6]. Within this group various approaches can be found, differing primarily in the weighting of harmonic peaks in the summation and the number of harmonics considered. Some systems also include a filtering step before the summation to exclude some spectral components based on energy and sinusoidality criteria [8] or spectral noise suppression [10].

Whilst the aforementioned systems have been compared in terms of melody extraction performance (c.f. MIREX), their overall complexity makes it hard to determine the effect of the first steps in each system on the final result. In this paper we aim to evaluate the first two processing steps (sinusoid extraction and salience function) alone, with the goal of understanding the benefits and caveats of the alternative approaches and how they might affect the rest of the system. Whilst some of these approaches have been compared in isolation before [9], our goal is to evaluate them under the specific context of melody extraction. For this purpose, a special evaluation framework, data-sets and metrics have been developed. In section 2 we described the different methods compared for extracting sinusoidal components, and in section 3 we describe the design of the salience function and the parameters affecting its computation. In section 4 we explain the evaluation framework used to evaluate both the sinusoid extraction and salience function design, together with the ground truth and metrics used. Finally, in section 5 we provide and discuss the results of the evaluation, summarised in the conclusions of section 6.

2. METHODS FOR SINUSOID EXTRACTION

The first step of many systems involves obtaining spectral components (peaks) from the audio signal, also referred to as the *front end* [7]. As mentioned earlier, different methods have been proposed to obtain the spectral peaks, usually with two common goals in mind – firstly, extracting the spectral peaks as accurately as possible in terms of their frequency and amplitude. Secondly, some systems attempt to enhance the amplitude of melody peaks whilst suppressing that of background peaks by applying some pre-filtering. For the purpose of our evaluation we have divided this process into three main steps, in each of which we consider two or three alternative approaches proposed in the literature. The alternatives considered at each step are summarised in Table 1.

* This research was funded by the Programa de Formación del Profesorado Universitario of the Ministerio de Educación de España, COFLA (P09-TIC-4840-JA) and DRIMS (TIN2009-14247-C02-01-MICINN).

¹http://www.music-ir.org/mirex/wiki/MIREX_HOME

Table 1: Analysis alternatives for sinusoid extraction.

Filtering	Spectral Transform	Frequency/Amplitude Correction
none Equal Loudness	STFT MRFFT	none Parabolic Interpolation Phase Vocoder

2.1. Filtering

As a first step, some systems filter the time signal in attempt to enhance parts of the spectrum more likely to pertain to the main melody, for example band-pass filtering [7]. For this evaluation we consider the more perceptually motivated equal loudness filtering. The equal loudness curves [11] describe the human perception of loudness as dependent on frequency. The equal loudness filter takes a representative average of these curves, and filters the signal by its inverse. In this way frequencies we are perceptually more sensitive to are enhanced in the signal, and frequencies we are less sensitive to are attenuated.

Further details about the implementation of the filter can be found here². It is worth noting that in the low frequency range the filter acts as a high pass filter with a high pass frequency of 150Hz. In our evaluation two alternatives are considered – equal loudness filtering, and no filtering³.

2.2. Spectral Transform

As previously mentioned, a potential problem with the STFT is that it has a fixed time and frequency resolution. When analysing an audio signal for melody extraction, it might be beneficial to have greater frequency resolution in the low frequencies where peaks are bunched closer together and are relatively stationary over time, and higher time resolution for the high frequencies where we can expect peaks to modulate rapidly over time (e.g. the harmonics of singing voice with a deep vibrato). In order to evaluate whether the use of a single versus multi-resolution transform is significant, two alternative transforms were implemented, as detailed below.

2.2.1. Short-Time Fourier Transform (Single Resolution)

The STFT can be defined as follows:

$$X_l(k) = \sum_{n=0}^{M-1} w(n) \cdot x(n + lH) e^{-j \frac{2\pi}{N} kn}, \quad (1)$$

$$l = 0, 1, \dots \text{ and } k = 0, 1, \dots, N-1$$

where $x(n)$ is the time signal, $w(n)$ the windowing function, l the frame number, M the window length, N the FFT length and H the hop size. We use the Hann windowing function with a window size of 46.4ms, a hop size of 2.9ms and a $\times 4$ zero padding factor. The evaluation data is sampled at $f_s = 44.1\text{kHz}$, giving $M = 2048$, $N = 8192$ and $H = 128$.

Given the FFT of a single frame $X(k)$, peaks are selected by finding all the local maxima k_m of the normalised magnitude spectrum $X_m(k)$:

²http://replaygain.hydrogenaudio.org/equal_loudness.html

³Spectral whitening/noise suppression is left for future work.

$$X_m(k) = 2 \frac{|X(k)|}{\sum_{n=0}^{M-1} w(n)}. \quad (2)$$

Peaks with a magnitude more than 80dB below the highest spectral peak in an excerpt are not considered.

2.2.2. Multi-Resolution FFT

We implemented the multi-resolution FFT (MRFFT) proposed in [8]. The MRFFT is an efficient algorithm for simultaneously computing the spectrum of a frame using different window sizes, thus allowing us to choose which window size to use depending on whether we require high frequency resolution (larger window size) or high time resolution (smaller window size). The algorithm is based on splitting the summations in the FFT into smaller sums which can be combined in different ways to form frames of varying sizes, and performing the windowing in the frequency domain by convolution. The resulting spectra all have the same FFT length N (i.e. smaller windows are zero padded) and use the Hann windowing function. For further details about the algorithm the reader is referred to [8].

In our implementation we set $N = 8192$ and $H = 128$ as with the STFT so that they are comparable. We compute four spectra $X_{256}(k)$, $X_{512}(k)$, $X_{1024}(k)$ and $X_{2048}(k)$ with respective window sizes of $M = 256, 512, 1024$ and 2048 samples (all windows are centered on the same sample). Then, local maxima (peaks) are found in each magnitude spectrum within a set frequency range as in [8], using the largest window (2048 samples) for the first six critical bands of the Bark scale (0-630Hz), the next window for the following five bands (630-1480Hz), the next one for the following five bands (1480-3150Hz) and the smallest window (256 samples) for the remaining bands (3150-22050Hz). The peaks from the different windows are combined to give a single set of peaks at positions k_m , and (as with the STFT) peaks with a magnitude more than 80dB below the highest peak in an excerpt are not considered.

2.3. Frequency and Amplitude Correction

Given the set of local maxima (peaks) k_m , the simplest approach for calculating the frequency and amplitude of each peak is to directly use its spectral bin and FFT magnitude (as detailed in equations 3 and 4 further down). This approach is limited by the frequency resolution of the FFT. For this reason various correction methods have been developed to achieve a higher frequency precision, and a better amplitude estimation as a result. In [12] a survey of these methods is provided for artificial, monophonic stationary sounds. Our goal is to perform a similar evaluation for real-world, polyphonic, quasi-stationary sounds (as is the case in melody extraction). For our evaluation we consider three of the methods discussed in [12], which represent three different underlying approaches:

2.3.1. Plain FFT with No Post-processing

Given a peak at bin k_m , its sine frequency and amplitude are calculated as follows:

$$\hat{f} = k_m \frac{f_s}{N} \quad (3)$$

$$\hat{a} = X_m(k_m) \quad (4)$$

Note that the frequency resolution is limited by the size of the FFT, in our case the frequency values are limited to multiples of $f_S/N = 5.38\text{Hz}$. This also results in errors in the amplitude estimation as it is quite likely for the true peak location to fall between two FFT bins, meaning the detected peak is actually lower (in magnitude) than the true magnitude of the sinusoidal component.

2.3.2. Parabolic Interpolation

This method improves the frequency and amplitude estimation of a peak by taking advantage of the fact that in the magnitude spectrum of most analysis windows (including the Hann window), the shape of the main lobe resembles a parabola in the dB scale. Thus, we can use the bin value and magnitude of the peak together with that of its neighbouring bins to estimate the position (in frequency) and amplitude of the true maximum of the main lobe, by fitting them to a parabola and finding its maximum. Given a peak at bin k_m , we define:

$$A_1 = X_{dB}(k_m - 1), A_2 = X_{dB}(k_m), A_3 = X_{dB}(k_m + 1), \quad (5)$$

where $X_{dB}(k) = 20 \log_{10}(X_m(k))$. The frequency difference in FFT bins between k_m and the true peak of the parabola is given by:

$$d = 0.5 \frac{A_1 - A_3}{A_1 - 2A_2 + A_3}. \quad (6)$$

The corrected peak frequency and amplitude (this time in dB) are thus given by:

$$\hat{f} = (k_m + d) \frac{f_S}{N} \quad (7)$$

$$\hat{a} = A_2 - \frac{d}{4} (A_1 - A_3) \quad (8)$$

Note that following the results of [12], the amplitude is not estimated using equation 8 above, but rather with equation 11 below, using the value of d as the bin offset $\kappa(k_m)$.

2.3.3. Instantaneous Frequency using Phase Vocoder

This approach uses the phase spectrum $\phi(k)$ to calculate the peak's instantaneous frequency (IF) and amplitude, which serve as a more accurate estimation of its true frequency and amplitude. The IF is computed from the phase difference $\Delta\phi(k)$ of successive phase spectra using the phase vocoder method [13] as follows:

$$\hat{f} = (k_m + \kappa(k_m)) \frac{f_S}{N}, \quad (9)$$

where the bin offset $\kappa(k)$ is calculated as:

$$\kappa(k) = \frac{N}{2\pi H} \Psi \left(\phi_l(k) - \phi_{l-1}(k) - \frac{2\pi H}{N} k \right), \quad (10)$$

where Ψ is the principal argument function which maps the phase to the $\pm\pi$ range.

The instantaneous magnitude is calculated using the peak's spectral magnitude $X_m(k_m)$ and the bin offset $\kappa(k_m)$ as follows:

$$\hat{a} = \frac{1}{2} \frac{X_m(k_m)}{W_{Hann} \left(\frac{M}{N} \kappa(k_m) \right)}, \quad (11)$$

where W_{Hann} is the Hann window kernel:

$$W_{Hann}(\kappa) = \frac{1}{2} \frac{\text{sinc}(\kappa)}{1 - \kappa^2}, \quad (12)$$

and sinc is the normalised sinc function. To achieve the best phase-based correction we use $H = 1$, by computing at each hop (of 128 samples) the spectrum of the current frame and of a frame shifted back by one sample, and using the phase difference between the two.

3. SALIENCE FUNCTION DESIGN

Once the spectral peaks are extracted, they are used to construct a salience function - a representation of frequency salience over time. For this study we use a common approach for salience computation based on harmonic summation, which was used as part of a complete melody extraction system in [6]. Basically, the salience of a given frequency is computed as the sum of the weighted energy of the spectral peaks found at integer multiples (harmonics) of the given frequency. As such, the important factors affecting the salience computation are the number of harmonics considered N_h and the weighting scheme used. In addition, we can add a relative magnitude filter, only considering for the summation peaks whose magnitude is no less than a certain threshold γ (in dB) below the magnitude of the highest peak in the frame. Note that the proposed salience function was designed as part of a system which handles octave errors and the selection of the melody pitch at a later stage, hence whilst the salience function is designed to best enhance melody salience compared to other pitched sources, these issues are not addressed directly by the salience function itself.

Our salience function covers a pitch range of nearly five octaves from 55Hz to 1.76kHz, quantized into $n = 1 \dots 600$ bins on a cent scale (10 cents per bin). Given a frequency f_i in Hz, its corresponding bin $b(f_i)$ is calculated as:

$$b(f_i) = \left\lfloor \frac{1200 \left(\log_2 \left(\frac{f_i}{13.75} \right) - 0.25 \right) - 2100}{10} + 1 \right\rfloor. \quad (13)$$

At each frame the salience function $S(n)$ is constructed using the spectral peaks p_i (with frequencies f_i and linear magnitudes m_i) found in the frame during the previous analysis step. The salience function is defined as:

$$S(n) = \sum_{h=1}^{N_h} \sum_{p_i} e(m_i) \cdot g(n, h, f_i) \cdot (m_i)^\beta, \quad (14)$$

where β is a parameter of the algorithm, $e(m_i)$ is a magnitude filter function, and $g(n, f_i, h)$ is the function that defines the weighting scheme. The magnitude filter function is defined as:

$$e(m_i) = \begin{cases} 1 & \text{if } 20 \log_{10}(m_M/m_i) < \gamma, \\ 0 & \text{otherwise,} \end{cases} \quad (15)$$

where m_M is the magnitude of the highest peak in the frame. The weighting function $g(n, f_i, h)$ defines the weight given to peak p_i , when it is considered as the h^{th} harmonic of bin n :

$$g(n, h, f_i) = \begin{cases} \cos^2(\delta \cdot \frac{\pi}{2}) \cdot \alpha^{h-1} & \text{if } |\delta| \leq 1, \\ 0 & \text{if } |\delta| > 1, \end{cases} \quad (16)$$

where $\delta = |b(f_i/h) - n|/10$ is the distance in semitones between the harmonic frequency f_i/h and the centre frequency of bin n and α is the harmonic weighting parameter. The threshold for δ means that each peak contributes not just to a single bin of the salience function but also to the bins around it (with \cos^2 weighting). This avoids potential problems that could arise due to the quantization of the salience function into bins, and also accounts for inharmonicities.

In sections 4 and 5 we will examine the effect of each of the aforementioned parameters on the salience function, in attempt to select a parameter combination most suitable for a salience function targeted at melody extraction. The parameters studied are the weighting parameters α and β , the magnitude threshold γ and the number of harmonics N_h used in the summation.

4. EVALUATION

The evaluation is split into two parts. First, we evaluate the different analysis approaches for extracting sinusoids in a similar way to [12]. The combination of different approaches at each step (filtering, transform, correction) gives rise to 12 possible analysis configurations, summarised in Table 2. In the second part, we evaluate the sinusoid extraction combined with the salience function computed using different parameter configurations. In the following sections we describe the experimental setup, ground truth and metrics used for each part of the evaluation.

Table 2: Analysis Configurations.

Conf.	Filtering	Spectral Transform	Frequency/Amplitude Correction
1	none	STFT	none
2			Parabolic
3			Phase
4		MRFFT	none
5			Parabolic
6			Phase
7	Eq. Loudness	STFT	none
8			Parabolic
9			Phase
10		MRFFT	none
11			Parabolic
12			Phase

4.1. Sinusoid Extraction

4.1.1. Ground Truth

Starting with a multi-track recording, the ground truth is generated by analysing the melody track on its own as in [14] to produce a per-frame list of f_0 + harmonics (up to the Nyquist frequency) with frequency and amplitude values. The output of the analysis is then re-synthesised using additive synthesis with linear frequency interpolation and mixed together with the rest of the tracks in the recording. The resulting mix is used for evaluating the different analysis configurations by extracting spectral peaks at every frame and comparing them to the ground truth. In this way we obtain a melody ground truth that corresponds perfectly to the melody

in the mixture, whilst being able to use real music as opposed to artificial mixtures.

As we are interested in the melody, only voiced frames are used for the evaluation (i.e. frames where the melody is present). Furthermore, some of the melody peaks will be masked in the mix by the spectrum of the accompaniment, where the degree of masking depends on the analysis configuration used. Peaks detected at frequencies where the melody is masked by the background depend on the background spectrum and hence should not be counted as successfully detected melody peaks. To account for this, we compute the spectra of the melody track and the background separately, using the analysis configuration being evaluated. We then check for each peak extracted from the mix by the analysis whether the melody spectrum is masked by the background spectrum at the peak frequency (a peak is considered to be masked if the spectral magnitude of the background is greater than that of the melody for the corresponding bin), and if so the peak is discarded.

The evaluation material is composed of excerpts from real-world recordings in various genres, summarised in Table 3.

Table 3: Ground Truth Material.

Genre	Excerpts	Tot. Melody Frames	Tot. Ground Truth Peaks
Opera	5	15,660	401,817
Pop/Rock	3	11,760	769,193
Instrumental Jazz	4	16,403	587,312
Bossa Nova	2	7,160	383,291

4.1.2. Metrics

We base our metrics on the ones used in [12], with some adjustments to account for the fact that we are only interested in the spectral peaks of the melody within a polyphonic mixture.

At each frame, we start by checking which peaks found by the algorithm correspond to peaks in the ground truth (melody peaks). A peak is considered a match if it is within 21.5Hz (equivalent to 1 FFT bin without zero padding) from the ground truth. If more than one match is found, we select the peak closest in amplitude to the ground truth. Once the matching peaks in all frames are identified, we compute the metrics R_p and R_e as detailed in Table 4.

Table 4: Metrics for sinusoid extraction.

R_p	Peak recall. The total number of melody peaks found by the algorithm in all frames divided by the total number of peaks in the ground truth.
R_e	Energy recall. The sum of the energy of all melody peaks found by the algorithm divided by the total energy of the peaks in the ground truth.
$\overline{\Delta a_{dB}}$	Mean amplitude error (in dB) of all detected melody peaks.
$\overline{\Delta f_c}$	Mean frequency error (in cents) of all detected melody peaks.
$\overline{\Delta f_w}$	Mean frequency error (in cents) of all detected melody peaks weighted by the normalised peak energy.

Given the matching melody peaks, we can compute the frequency estimation error Δf_c and the amplitude estimation error Δa_{dB} of each peak⁴. The errors are measured in cents and dBs respectively, and averaged over all peaks of all frames to give $\overline{\Delta f_c}$ and $\overline{\Delta a_{dB}}$. A potential problem with $\overline{\Delta f_c}$ is that the mean may be dominated by peaks with very little energy (especially at high frequencies), even though their effect on the harmonic summation later on will be insignificant. For this reason we define a third measure $\overline{\Delta f_w}$, which is the mean frequency error in cents where each peak's contribution is weighted by its energy, normalised by the energy of the highest peak in the ground truth in the same frame. The normalisation ensures the weighting is independent of the volume of each excerpt⁵. The metrics are summarised above in Table 4.

4.2. Saliency Function Design

In the second part of the evaluation we take the spectral peaks produced by each one of the 12 analysis configurations and use them to compute the saliency function with different parameter configurations. The saliency function is then evaluated in terms of its usefulness for melody extraction using the ground truth and metrics detailed below.

4.2.1. Ground Truth

We use the same evaluation material as in the previous part of the evaluation. The first spectral peak in every row of the ground truth represents the melody f0, and is used to evaluate the frequency accuracy of the saliency function as explained below.

4.2.2. Metrics

We evaluate the saliency function in terms of two aspects – frequency accuracy and melody saliency, where melody saliency should reflect the predominance of the melody compared to the other pitched elements appearing in the saliency function. Four metrics have been devised for this purpose, computed on a per-frame basis and finally averaged over all frames.

We start by selecting the peaks of the saliency function. The saliency peak closest in frequency to the ground truth f0 is considered the melody saliency peak. We can then calculate the frequency error of the saliency function Δf_m as the difference in cents between the frequency of the melody saliency peak and the ground truth f0.

To evaluate the predominance of the melody three metrics are computed. The first is the rank R_m of the melody saliency peak amongst all saliency peaks in the frame, which ideally should be 1. Rather than report the rank directly we compute the reciprocal rank $RR_m = 1/R_m$ which is less sensitive to outliers when computing the mean over all frames. The second is the relative saliency S_1 of the melody peak, computed by dividing the saliency of the melody peak by that of the highest peak in the frame. The third metric, S_3 , is the same as the previous one only this time we divide the saliency of the melody peak by the mean saliency of the top 3 peaks of the saliency function. In this way we can measure not only

whether the melody saliency peak is the highest, but also whether it stands out from the other peaks of the saliency function and by how much. The metrics are summarised in Table 5.

Table 5: Metrics for evaluating Saliency Function Design.

Δf_m	Melody frequency error.
RR_m	Reciprocal Rank of the melody saliency peak amongst all peaks of the saliency function.
S_1	Melody saliency compared to top peak.
S_3	Melody saliency compared to top 3 peaks.

5. RESULTS

The results are presented in two stages. First we present the results for the sinusoid extraction, and then the results for the saliency function design. In both sections, each metric is evaluated for each of the 12 possible analysis configurations summarised in Table 2.

5.1. Sinusoid Extraction

We start by examining the results obtained when averaging over all genres, provided in Table 6. The best result in each column is highlighted in bold. Recall that R_p and R_e should be maximised whilst Δa_{dB} , $\overline{\Delta f_c}$ and $\overline{\Delta f_w}$ should be minimised.

Table 6: Sinusoid extraction results for all genres.

Conf.	R_p	R_e	Δa_{dB}	$\overline{\Delta f_c}$	$\overline{\Delta f_w}$
1	0.62	0.88	3.03	3.17	8.77
2	0.62	0.88	3.02	2.89	7.20
3	0.62	0.88	3.02	2.88	6.91
4	0.29	0.84	1.43	5.21	9.60
5	0.29	0.84	1.43	4.75	7.99
6	0.31	0.85	1.46	4.35	7.40
7	0.55	0.88	2.79	3.47	8.10
8	0.55	0.88	2.78	3.16	6.69
9	0.54	0.88	2.78	3.13	6.45
10	0.27	0.83	1.41	5.63	9.04
11	0.27	0.83	1.41	5.13	7.58
12	0.27	0.84	1.45	4.84	7.03

We see that regardless of the filtering and transform used, both parabolic and phase based correction provide an improvement in frequency accuracy (i.e. lower $\overline{\Delta f_c}$ values), with the phase based method providing just slightly better results. The benefit of using frequency correction is further accentuated when considering $\overline{\Delta f_w}$. As expected, there is no significant difference between the amplitude error Δa_{dB} when correction is applied and when it is not, as the error is dominated by the spectrum of the background.

When considering the difference between using the STFT and MRFFT, we first note that there is no significant improvement in frequency accuracy (i.e. smaller frequency error) when using the MRFFT (for all correction options), as indicated by both $\overline{\Delta f_c}$ and $\overline{\Delta f_w}$. This suggests that whilst the MRFFT might be advantageous for certain types of data (c.f. results for opera in Table 7), when averaged over all genres the method does not provide a significant improvement in frequency accuracy.

⁴As we are using polyphonic material the amplitude error may not reflect the accuracy of the method being evaluated, and is included for completeness.

⁵Other weighting schemes were tested and shown to produce very similar results.

When we turn to examine the peak and energy recall, we see that the STFT analysis finds more melody peaks, however, interestingly both transforms obtain a similar degree of energy recall. This implies that the MRFFT, which generally finds less peaks (due to masking caused by wider peak lobes), still finds the most important melody peaks. Whether this is significant or not for melody extraction should become clearer in the second part of the evaluation when examining the salience function.

Next, we observe the effect of applying the equal loudness filter. We see that peak recall is significantly reduced, but that energy recall is maintained. This implies that the filter does not attenuate the most important melody peaks. If, in addition, the filter attenuates some background peaks, the overall effect would be that of enhancing the melody. As with the spectral transform, the significance of this step will become clearer when evaluating the salience function.

Finally, we provide the results obtained for each genre separately in Table 7 (for brevity only configurations which obtain the best result for at least one of the metrics are included). We can see that the above observations hold for the individual genres as well. The only interesting difference is that for the opera genre the MRFFT gives slightly better overall results compared to the STFT. This can be explained by the greater pitch range and deep vibrato which often characterise the singing in this genre. The MRFFT's increased time resolution at higher frequencies means it is better at estimating the rapidly changing harmonics present in opera singing.

Table 7: Sinusoid extraction results per genre.

Genre	Conf.	R_p	R_e	Δa_{dB}	Δf_c	Δf_w
Opera	2	0.73	0.83	3.74	3.97	7.48
	6	0.59	0.93	1.15	3.66	6.50
	11	0.53	0.92	1.08	3.88	5.91
Jazz	3	0.57	0.96	2.20	2.33	6.23
	9	0.56	0.96	2.18	2.36	5.75
	10	0.20	0.84	1.57	7.88	10.95
Pop/Rock	2	0.54	0.84	3.08	3.05	7.71
	3	0.54	0.83	3.08	3.05	7.43
	9	0.46	0.84	2.89	3.37	6.83
	11	0.17	0.73	1.86	6.73	8.97
Bossa Nova	2	0.76	0.91	3.17	1.95	5.75
	8	0.56	0.92	2.74	2.32	5.48
	9	0.56	0.92	2.74	2.36	5.30
	10	0.29	0.86	1.33	4.19	8.00

5.2. Salience Function Design

As explained in section 3, in addition to the analysis configuration used, the salience function is determined by four main parameters – the weighting parameters α and β , the energy threshold γ and the number of harmonics N_h . To find the best parameter combination for each analysis configuration and to study the interaction between the parameters, we performed a grid search of these four parameters using several representative values for each parameter: $\alpha = 1, 0.9, 0.8, 0.6$, $\beta = 1, 2$, $\gamma = \infty, 60\text{dB}, 40\text{dB}, 20\text{dB}$, and $N_h = 4, 8, 12, 20$. This results in 128 possible parameter combinations which were used to compute the salience function metrics for each of the 12 analysis configurations.

We started by plotting a graph for each metric with a data point for each of the 128 parameter combinations, for the 12 analysis

configurations⁶. At first glance it was evident that for all analysis and parameter configurations the results were consistently better when $\beta = 1$, thus only the 64 parameter configurations in which $\beta = 1$ shall be considered henceforth.

5.2.1. Analysis Configuration

We start by examining the effect of the analysis configuration on the salience function. In Figure 1 we plot the results obtained for each metric by each configuration. For comparability the salience function is computed using the same (optimal) parameter values ($\alpha = 0.8$, $\beta = 1$, $\gamma = 40\text{dB}$, $N_h = 20$) for all analysis configurations (the parameter values are discussed in section 5.2.2). Configurations that only differ in the filtering step are plotted side by side. Metrics Δf_m , RR_m , S_1 and S_3 are displayed in plots (a), (b), (c) and (d) of Figure 1 respectively.

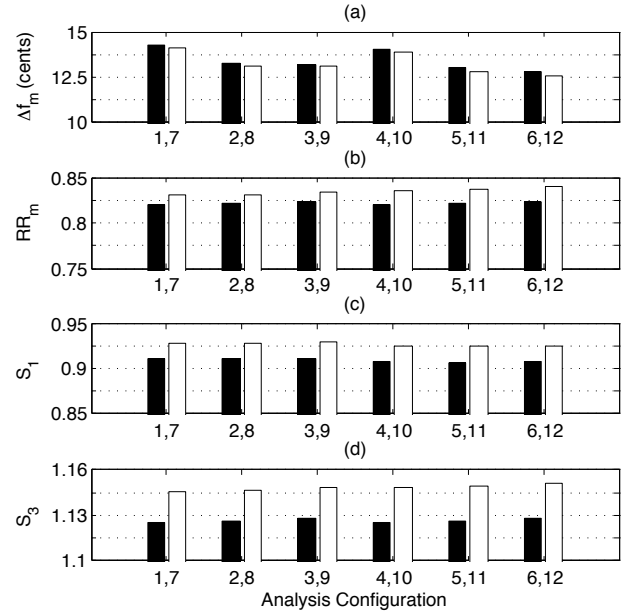


Figure 1: Salience function design, overall results. Each bar represents an analysis configuration, where white bars are configurations which apply equal loudness filtering. Recall that Δf_m should be minimised whilst RR_m , S_1 and S_3 should be maximised.

The first thing we see is that for all metrics, results are always improved when equal loudness filtering is applied. This confirms our previous stipulation that the filter enhances the melody by attenuating non-melody spectral peaks. It can be explained by the filter's enhancement of the mid-band frequencies which is where the melody is usually present, and the attenuation of low-band frequencies where we expect to find low pitched instruments such as the bass.

Next we examine the frequency error Δf_m in Figure 1 plot (a). We see that there is a (significant) decrease in the error when either of the two correction methods (parabolic interpolation or phase vocoder) are applied, as evident by comparing configurations 1, 7, 4, 10 (no correction) to the others. Though the error

⁶For brevity these plots are not reproduced in the article but can be found at: <http://mtg.upf.edu/node/2023>.

using phase based correction is slightly lower, the difference between the two correction methods was not significant. Following these observations, we can conclude that both equal loudness filtering and frequency correction are beneficial for melody extraction.

Finally we consider the difference between the spectral transforms. Interestingly, the MRFFT now results in just a slightly lower frequency error than the STFT. Whilst determining the exact cause is beyond the scope of this study, a possible explanation could be that whilst the overall frequency accuracy for melody spectral peaks is not improved by the MRFFT, the improved estimation at high frequencies is beneficial when we do the harmonic summation (the harmonics are better aligned). Another possible cause is the greater masking of spectral peaks, which could remove non-melody peaks interfering with the summation. When considering the remaining metrics, the STFT gives slightly better results for S_1 , whilst there is no statistically significant difference between the transforms for RR_m and S_3 . All in all, we see that using a multi-resolution transform provides only a marginal improvement (less than 0.5 cents) in terms of melody frequency accuracy, suggesting it might not necessarily provide significantly better results in a complete melody extraction system.

5.2.2. Saliency Function Parameter Configuration

We now turn to evaluate the effect of the parameters of the saliency function. In the previous section we saw that equal loudness filtering and frequency correction are important, whilst the type of correction and transform used do not affect the results significantly. Thus, in this section we will focus on configuration 9, which applies equal loudness filtering and uses the STFT transform with phase vocoder frequency correction⁷.

In Figure 2 we plot the results obtained for the four metrics using configuration 9 with each of the 64 possible parameter configurations ($\beta = 1$ in all cases) for the saliency function. The first 16 datapoints represent configurations where $\alpha = 1$, the next 16 where $\alpha = 0.9$ and so on. Within each group of 16, the first 4 have $N_h = 4$, the next 4 have $N_h = 8$ etc. Finally within each group of 4, each datapoint has a different γ value from ∞ down to 20dB.

We first examine the effect of the peak energy threshold γ , by comparing individual datapoints within every group of 4 (e.g. comparing peaks 1-4, 29-32 etc.). We see that (for all metrics) there is no significant difference for the different values of the threshold except for when it is set to 20dB for which the results degrade. That is, unless the filtering is too strict, filtering relatively weak spectral peaks seems to neither improve nor degrade the results.

Next we examine the effect of N_h , by comparing different groups of 4 data points within every group of 16 (e.g. 17-20 vs 25-28). With the exception of the configurations where $\alpha = 1$ (1-16), for all other configurations all metrics are improved the more harmonics we consider. As the melody in our evaluation material is primarily human voice (which tends to have many harmonic partials), this makes sense. We can explain the decrease for configurations 1-16 by the lack of harmonic weighting ($\alpha = 1$) which results in a great number of fake peaks with high salience at integer/sub-integer multiples of the true f_0 .

Finally, we examine the effect of the harmonic weighting parameter α . Though it has a slight effect on the frequency error, we are primarily interested in its effect on melody salience as indicated by RR_m , S_1 and S_3 . For all three metrics, no weighting (i.e. $\alpha = 1$) never produces the best results. For RR_m and S_1 we

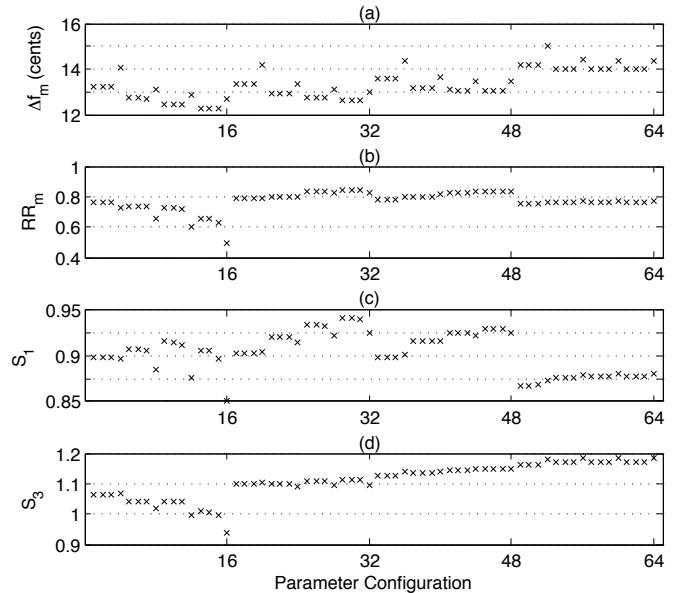


Figure 2: Saliency function design, results by parameter configuration.

get best performance when α is between 0.9 and 0.8. Interestingly, S_3 increases continually as we decrease α . This implies that even with weighting, fake peaks at integer/sub-integer multiples (which are strongly affected by α) are present. This means that regardless of the configuration used, systems which use saliency functions based on harmonic summation should include a post-processing step to detect and discard octave errors.

In Figure 3 we plot the metrics as a function of the parameter configuration once more, this time for each genre (using analysis configuration 9). Interestingly, opera, jazz and bossa nova behave quite similarly to each other and to the overall results. For pop/rock however we generally get slightly lower results, and there is greater sensitivity to the parameter values. This is most likely due to the fact that the accompaniment is more predominant in this genre, making it harder for the melody to stand out. In this case we can expect to find more predominant peaks in the saliency function which represent background instruments rather than octave errors of the melody. Consequently, S_3 no longer favours the lowest harmonic weighting and, like RR_m and S_1 , gives best results for $\alpha = 0.8$ or 0.9.

Following the above analysis, we can identify the combination of saliency function parameters that gives the best overall results across all four metrics as $\alpha = 0.8$ or 0.9, $\beta = 1$, $N_h = 20$ and $\gamma = 40$ dB or higher.

6. CONCLUSIONS

In this paper the first two steps common to a large group of melody extraction systems were studied - sinusoid extraction and saliency function design. Several analysis methods were compared for sinusoid extraction and it was shown that accuracy is improved when frequency/amplitude correction is applied. Two spectral transforms (single and multi-resolution) were compared and shown to perform similarly in terms of melody energy recall and frequency accuracy.

⁷Configurations 8, 11 and 12 result in similar graphs.

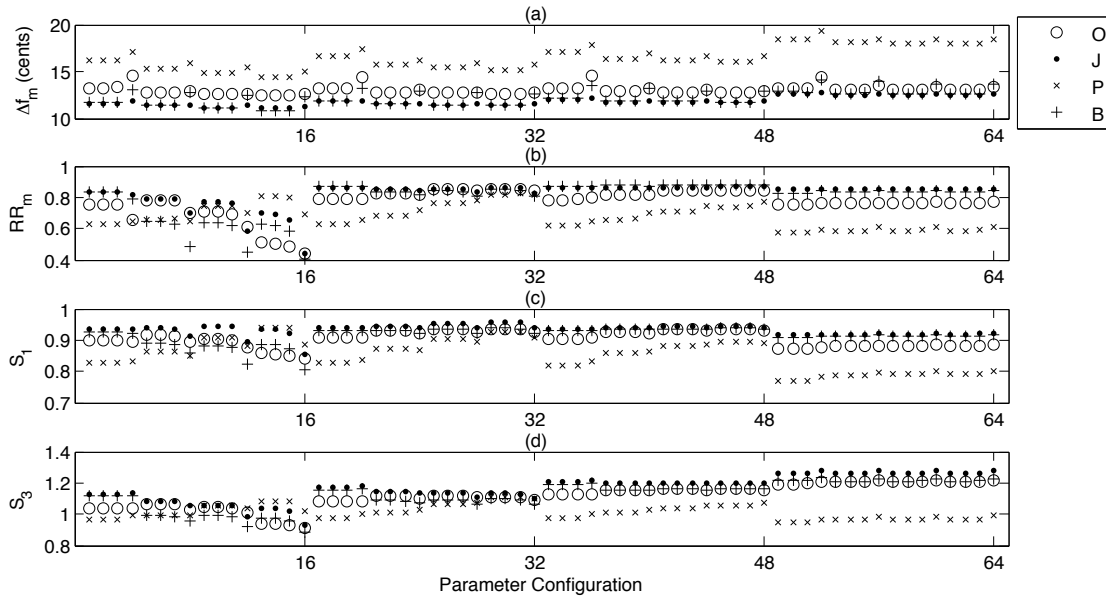


Figure 3: Per genre results by parameter configuration. Genres are labeled by their first letter – Opera, Jazz, Pop/Rock and Bossa Nova.

A salience function based on harmonic summation was introduced alongside its key parameters. The different analysis configurations were all evaluated in terms of the salience function they produce, and the effects of the parameters on the salience function were studied. It was shown that equal loudness and frequency correction both result in significant improvements to the salience function, whilst the difference between the alternative frequency correction methods or the single/multi-resolution transforms was marginal. The effect of the different parameters on the salience function was studied and an overall optimal analysis and parameter configuration for melody extraction using the proposed salience function was identified.

7. ACKNOWLEDGMENTS

The authors would like to thank Ricard Marxer, Perfecto Herrera, Joan Serrà and Martín Haro for their comments.

8. REFERENCES

- [1] G. E. Poliner, D. P. W. Ellis, F. Ehmann, E. Gómez, S. Steich, and B. Ong, “Melody transcription from music audio: Approaches and evaluation,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 4, pp. 1247–1256, 2007.
- [2] Jean-Louis Durrieu, Gaël Richard, Bertrand David, and Cédric Févotte, “Source/filter model for unsupervised main melody extraction from polyphonic audio signals,” *Trans. Audio, Speech and Lang. Proc.*, vol. 18, pp. 564–575, 2010.
- [3] M. Ryyänen and A. Klapuri, “Automatic transcription of melody, bass line, and chords in polyphonic music,” *Computer Music Journal*, vol. 32, no. 3, pp. 72–86, 2008.
- [4] P. Cancela, “Tracking melody in polyphonic audio,” in *4th Music Information Retrieval Evaluation eXchange (MIREX)*, 2008.
- [5] Karin Dressler, “Audio melody extraction for mirex 2009,” in *5th Music Information Retrieval Evaluation eXchange (MIREX)*, 2009.
- [6] J. Salamon and E. Gómez, “Melody extraction from polyphonic music audio,” in *6th Music Information Retrieval Evaluation eXchange (MIREX)*, extended abstract, 2010.
- [7] M. Goto, “A real-time music-scene-description system: predominant-f0 estimation for detecting melody and bass lines in real-world audio signals,” *Speech Communication*, vol. 43, pp. 311–329, 2004.
- [8] K. Dressler, “Sinusoidal extraction using an efficient implementation of a multi-resolution FFT,” in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 2006, pp. 247–252.
- [9] P. Cancela, M. Rocamora, and E. López, “An Efficient Multi-Resolution Spectral Transform for Music Analysis,” in *Proc. of the 10th Int. Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 2009, pp. 309–314.
- [10] A. P. Klapuri, “Multiple Fundamental Frequency Estimation based on Harmonicity and Spectral Smoothness,” in *IEEE Trans. Speech and Audio Processing*, 2003, vol. 11.
- [11] D. W. Robinson and R. S. Dadson, “A re-determination of the equal-loudness relations for pure tones,” *British Journal of Applied Physics*, vol. 7, pp. 166–181, 1956.
- [12] Florian Keiler and Sylvain Marchand, “Survey on extraction of sinusoids in stationary sounds,” in *Proc. of the 5th Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, Sept. 2002, pp. 51–58.
- [13] J. L. Flanagan and R. M. Golden, “Phase vocoder,” *Bell Systems Technical Journal*, vol. 45, pp. 1493–1509, 1966.
- [14] J. Bonada, “Wide-band harmonic sinusoidal modeling,” in *Proc. 11th Int. Conf. on Digital Audio Effects (DAFX-08)*, Espoo, Finland, Sept. 2008.

SPARSE ATOMIC MODELING OF AUDIO: A REVIEW

Corey Kereliuk,

SPCL* & CIRMMT†

McGill University, Montréal, Canada

corey.kereliuk@mail.mcgill.ca

Philippe Depalle,

SPCL* & CIRMMT†

McGill University, Montréal, Canada

depalle@music.mcgill.ca

ABSTRACT

Research into sparse atomic models has recently intensified in the image and audio processing communities. While other reviews exist, we believe this paper provides a good starting point for the uninitiated reader as it concisely summarizes the state-of-the-art, and presents most of the major topics in an accessible manner. We discuss several approaches to the sparse approximation problem including various greedy algorithms, iteratively re-weighted least squares, iterative shrinkage, and Bayesian methods. We provide pseudo-code for several of the algorithms, and have released software which includes fast dictionaries and reference implementations for many of the algorithms. We discuss the relevance of the different approaches for audio applications, and include numerical comparisons. We also illustrate several audio applications of sparse atomic modeling.

1. INTRODUCTION

Many natural signals can be sparsely represented (or sparsely approximated) if an appropriate basis can be found. For example, a short block of samples from a quasi-periodic sound will have a sparse Fourier transform if the block size is a multiple of the pitch period. We often seek sparse representations, or sparse models, because they lead to a clear interpretation. If we compare several models that summarize a data set equally well, we usually prefer the sparser models, since each variable tends to be more meaningful¹. This is especially true if we are interested in audio effects, since we desire a meaningful mapping between the control parameters and the perceived outcome.

In this paper we limit ourselves to the following model:

$$\mathbf{y} = \Phi \mathbf{x} + \varepsilon \quad (1)$$

where $\mathbf{y} \in \mathbb{R}^M$ is a sampled sound, $\Phi \in \mathbb{C}^{M \times N}$ is a dictionary of (possibly) complex atoms, and ε is additive noise. We call $\text{supp}(\mathbf{x}) = \{i | x_i \neq 0\}$ the support of \mathbf{x} , and define the sparsity of \mathbf{x} as $\|\mathbf{x}\|_0 = |\text{supp}(\mathbf{x})|$, which is the cardinality of the support. We say that \mathbf{y} is synthesis-sparse in the dictionary Φ if $\|\mathbf{x}\|_0 \ll M$. In this paper we focus primarily on synthesis sparsity, however, it is worth noting that several recent works consider signals which are analysis-sparse: that is, signals for which $\|\Omega \mathbf{x}\|_0 \ll M$ (where Ω is an analysis operator) [1, 2].

Although, real sound signals may not be truly sparse in any basis, they are often *compressible*. We say that \mathbf{y} is compressible in Φ if the sorted magnitudes of \mathbf{x} decay according to a power-law.

This means that we may discard many of the small coefficients in \mathbf{x} without a huge sacrifice in the perceived quality.

The formulation in (1) is quite common in audio processing since we may consider the wavelet transform, the modified discrete cosine transform (MDCT), and the short time Fourier transform (STFT) as instances of this model (if we choose Φ appropriately and set ε to 0).

In this paper we focus on the case where $N > M$, which means that the dictionary contains a redundant set of waveforms. This situation arises quite naturally in audio processing. For example, the STFT is often oversampled so that *i*) smoother analysis windows may be used, and *ii*) to make the transform more invariant to shifts in the input signal. Both of these considerations lead to a redundant dictionary. Furthermore, it is often useful to build hybrid dictionaries from the union of several different dictionaries. This allows us to match the dictionary waveforms to the type of features we expect to encounter in the signal. As described in §10.1 this fact can be used to build multilayer signal expansions.

In the first part of this paper we examine several state-of-the-art approaches for estimating a sparse \mathbf{x} given a redundant dictionary Φ . We discuss most of the major approaches and their variants. Along the way we point out which algorithms have the potential to work with the large data sets common in audio applications. In the second part of this paper we perform some numerical comparisons of these algorithms, and review some important audio applications that can benefit from sparse atomic modeling.

2. THE METHOD OF FRAMES

We first consider the case without an explicit noise term, i.e., $\mathbf{y} = \Phi \mathbf{x}$. There are many possible solutions that satisfy this equation when $N > M$ since Φ has a null space (and adding an element from the null space does not change the solution). One possible solution is:

$$\mathbf{x} = \Phi^H \mathbf{S}^{-1} \mathbf{y} \quad (2)$$

where $\mathbf{S} = \Phi \Phi^H$ is called the frame operator, and Φ^H denotes the conjugate transpose. The frame operator is invertible if its minimum eigenvalue is greater than zero and its maximum eigenvalue is finite. In the finite dimensional case (which is the only case we consider in this paper), the latter condition is always satisfied, and the former condition is satisfied whenever Φ has rank M (which is to say the columns of Φ span M dimensional space). In the literature (2) is known as the method of frames (MOF) [3]. A very comprehensive review on frames can be found in [4].

The MOF solution is unique in the sense that \mathbf{x} is orthogonal to the null space of Φ and hence has the minimum 2-norm out of all possible solutions. As such the MOF can be viewed as the

*Sound Processing and Control Laboratory

†Centre for Interdisciplinary Research in Music Media and Technology

¹This principle is often referred to as *Occam's razor*.

solution to the following problem:

$$\arg \min_{\mathbf{x}} \frac{1}{2} \|\mathbf{x}\|_2^2 \quad \text{subject to} \quad \mathbf{y} = \Phi \mathbf{x} \quad (3)$$

As noted by several authors the MOF solution is usually *not sparse* due to the fact that the 2-norm places a very high penalty on large coefficients (and thus there tend to be many small yet significant coefficients) [5].

3. SPARSE APPROXIMATIONS

As noted in the introduction the ℓ_0 pseudo norm, $\|\mathbf{x}\|_0$, is a direct measure of sparsity. In light of the MOF formulation in (3) this leads us to question if we can solve the following problem:

$$\arg \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \Phi \mathbf{x}\|_2^2 \leq T \quad (4)$$

where T is proportional to the noise variance (or 0 in the noiseless case). Unfortunately, the solution to this problem involves the enumeration of all possible subsets of columns from Φ , which is a combinatorial problem [6]. This problem is also unstable in the noiseless case, since a small perturbation of \mathbf{y} can dramatically effect $\|\mathbf{x}\|_0$.

There are two main strategies that have been explored in the literature to recover sparse solutions. The first tactic is to use greedy algorithms, which build up a solution by selecting one coefficient in \mathbf{x} per iteration. By stopping a greedy algorithm early, a sparse approximation is guaranteed. The second strategy is to rephrase (4) using a cost function that can be tractably minimized. We refer to this approach as relaxation. Although both of these tactics often lead to suboptimal solutions, there are certain conditions under which the optimal solution may be recovered [7, 8].

In the following sections we explore these two approaches and discuss algorithms for their solutions. We then provide some numerical results to compare the different algorithms, and discuss their suitability for audio applications.

4. GREEDY APPROACHES

4.1. Matching Pursuit

The matching pursuit (MP) algorithm is one of the most well known greedy algorithms for sparse approximation [9]. In MP we start with an empty solution $\mathbf{x}^{(0)} = \mathbf{0}$, and adjust one coefficient in \mathbf{x} at each iteration. This coefficient is chosen so as to minimize the residual error at each iteration. For example, on the n^{th} iteration we define $\mathbf{x}^{(n)} = \mathbf{x}^{(n-1)} + \alpha \delta$, where α is a scalar and δ is a unit vector with one non-zero component that indicates which element of \mathbf{x} should be updated. The residual can then be written as $\mathbf{r}^{(n)} = \mathbf{y} - \Phi \mathbf{x}^{(n)} = \mathbf{r}^{(n-1)} - \alpha \phi$, where the atom ϕ is the column of Φ identified by δ . At each iteration we seek an atom ϕ and scalar α that minimize the current residual:

$$\arg \min_{\alpha \in \mathbb{C}, \phi \in \Phi} \frac{1}{2} \|\mathbf{r}^{(n-1)} - \alpha \phi\|_2^2 \quad (5)$$

Solving for α we find

$$\alpha = \frac{\phi^H \mathbf{r}^{(n-1)}}{\phi^H \phi} \quad (6)$$

where $\phi^H \mathbf{r}^{(n-1)} = \sum_k \phi^*[k] \mathbf{r}^{(n-1)}[k]$. We often normalize the dictionary atoms so that $\phi^H \phi = 1$ (we will assume this is the

case from here on out). Plugging this value of α back into (5) it is straightforward to show that the atom which decreases the residual error most is given by

$$\arg \max_{\phi \in \Phi} |\phi^H \mathbf{r}^{(n-1)}| \quad (7)$$

Algorithm 1 summarizes the steps in MP.

Algorithm 1 Matching Pursuit

```

1: init:  $n = 0, \mathbf{x}^{(n)} = \mathbf{0}, \mathbf{r}^{(n)} = \mathbf{y}$ 
2: repeat
3:    $i_n = \arg \max_i |\phi_i^H \mathbf{r}^{(n)}|$ 
4:    $\alpha_n = \phi_{i_n}^H \mathbf{r}^{(n)}$ 
5:    $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \alpha_n \delta_{i_n}$ 
6:    $\mathbf{r}^{(n+1)} = \mathbf{r}^{(n)} - \alpha_n \phi_{i_n}$ 
7:    $n = n + 1$ 
8: until stopping condition

```

The stopping condition is usually based on a combination of the desired signal to residual ratio (SRR), and maximum number of iterations allowed.

After k iterations the signal approximation is

$$\hat{\mathbf{y}} = \sum_{n=0}^{k-1} \alpha_n \phi_{i_n} \quad (8)$$

We can avoid explicit computation of the residual in algorithm 1 if we multiply both sides of line 6 by ϕ_j^H . This gives

$$\phi_j^H \mathbf{r}^{(n+1)} = \phi_j^H \mathbf{r}^{(n)} - \alpha_n \mathbf{G}[i_n, j] \quad (9)$$

where $\mathbf{G} = \Phi^H \Phi$ is the Gram matrix, and $\phi_j^H \mathbf{r}^{(n)}$ was already calculated in the previous iteration. In practice the Gram matrix is often too big to be stored, however, in many cases it will have a sparse structure so an update of this form can still be useful. For example, when local dictionaries are used the majority of entries in the Gram matrix are zero, so many of the inner products do not need to be updated [10].

Equation (9) reveals that the inner products at iteration $n + 1$ depend on the atom selected at iteration n (via the Gram matrix). When the atoms are correlated (as they will be in a redundant dictionary) this dependence can lead to the algorithm making suboptimal choices. Nonetheless, the residual is guaranteed to converge to zero in norm as the number of iterations tends to infinity [9].

4.2. Variants and Extensions

It should be noted that in MP we may select atoms at each iteration based on criteria other than minimizing the residual energy. For example, if we have some *a priori* knowledge about the signal we can modify the selection criteria to include this information. To illustrate, in [11] a psychoacoustic weighting was applied before minimizing the residual, and in [12] an MP-variant was introduced that avoids selecting atoms which might lead to pre-echo artifacts. Likewise, we might also restrict the search region for atoms at each iteration. For example, in [13] the search region was restricted so that only overlapping chains of atoms (similar to partials) were extracted by the algorithm. This flexibility in the selection of atoms is a great advantage of MP over some of the other algorithms introduced later.

With MP we can also refine the atom parameters at each iteration using Newton's method [9]. This allows one to find a continuous estimate for the atom parameters even when using a discrete dictionary. Also, in [14] a method known as cyclic matching pursuit was introduced that allows one to find a continuous estimate of the amplitude and frequency parameters when using a dictionary of complex sinusoids.

4.3. Orthogonalization

There are several variants of MP which use orthogonalization to improve the performance (i.e., achieve a higher SRR with fewer atoms). For example, we can update the coefficient vector \mathbf{x} by orthogonal projection every k iterations. This process is known as backprojection. If we let $\Delta_k \triangleq \text{supp}(\mathbf{x}^{(k)})$, and denote $\mathbf{x}_{\Delta_k} \triangleq \{x_i | i \in \Delta_k\}$ and $\Phi_{\Delta_k} \triangleq \cup_{i \in \Delta_k} \phi_i$, then we can write backprojection as:

$$\hat{\mathbf{x}}_{\Delta_k} = \arg \min_{\text{supp}(\mathbf{x})=\Delta_k} \frac{1}{2} \|\mathbf{y} - \Phi_{\Delta_k} \mathbf{x}\|_2^2 \quad (10)$$

The solution to this equation is given by $\hat{\mathbf{x}}_{\Delta_k} = \Phi_{\Delta_k}^+ \mathbf{y}$, where $\Phi_{\Delta_k}^+$ indicates the pseudo-inverse.

If we carry backprojection to its logical extreme, and update the coefficients after every iteration we arrive at an MP-variant known as orthogonal matching pursuit (OMP) [15]. OMP tends to be very computationally expensive, since it requires computation and inversion of the partial Gram matrix at every iteration. However, since the residual remains orthogonal to the selected atoms at every iteration, OMP will never select the same atom twice (this is not the case for MP), and is guaranteed to converge in M (or fewer) iterations.

There are also several variants of OMP that have been discussed in the literature. For example, in optimized OMP (OOMP) [16] the atom selection metric is adjusted in order to improve the residual decay, and in stagewise OMP (StOMP) [17] several atoms are selected and orthogonalized at each iteration.

In [18] several fast algorithms were introduced which approximate OMP using gradient and conjugate gradient information. Further, in [10] a fast approximate OMP algorithm was proposed for use with local dictionaries. This algorithm exploits the fact that many dictionaries of practical interest are local in the sense that the majority of atoms are orthogonal to one another (since they are supported on disjoint sets). This allows one to work with a much smaller partial Gram matrix, and dramatically speeds up the algorithm in practice. As such these algorithms hold considerable promise for audio applications.

4.4. Conjugate Subspaces

In standard MP we select just one atom at each iteration. When working with complex atoms and a real signal it can be useful to select a conjugate subspace at each iteration. This can be done by replacing α by $[\alpha \ \alpha^*]^T$ and ϕ by $[\phi \ \phi^*]$ in (5) which leads to the solution outlined in [5].

Using complex atoms with conjugate subspaces leads to two important advantages when working with audio. Firstly, using complex atoms allows one to estimate the phase without explicitly parameterizing this value. Second, when selecting low or high frequency atoms the inner products can be biased by spectral leakage from the negative frequency spectrum. Since this approach

selects a subspace consisting of one positive and one negative frequency atom, it is resilient to this possible bias (further details can be found in [19]).

4.5. Weak Matching Pursuit

A modification to MP known as weak matching pursuit (WMP) can be practically useful when dealing with very large dictionaries, where the computation of inner products would ordinarily be prohibitive [9, 20]. At each iteration of WMP we select an atom from a subset of the full dictionary:

$$\Phi_{\Lambda}^{(n)} = \left\{ \phi_i \left| \left| \phi_i^H \mathbf{r}^{(n)} \right| \geq \beta \max_j \left| \phi_j^H \mathbf{r}^{(n)} \right| \right\} \quad (11)$$

where $\beta \in (0, 1]$ is a relaxation factor. It has been shown that the WMP will converge even if β changes from iteration to iteration [21]. In [22] and [23] this strategy was used to prune the dictionary around local maxima, leading to a significant computational savings.

5. RELAXED APPROACHES

As mentioned in §3 the second major class of algorithms for sparse approximation are based on relaxation. In essence, we relax the hard ℓ_0 pseudo norm problem by replacing it with a cost function that can be tractably minimized.

In order to proceed let us replace the ℓ_0 pseudo norm in (4) by a function $f(\mathbf{x})$ that measures the sparsity of \mathbf{x} :

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad \|\mathbf{y} - \Phi \mathbf{x}\|_2^2 \leq T \quad (12)$$

This equation can also be written in an equivalent unconstrained form:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - \Phi \mathbf{x}\|_2^2 + \lambda f(\mathbf{x}) \quad (13)$$

We refer to $f(\mathbf{x})$ as a regularization term, and note that the scalar λ controls the degree of regularization (i.e., trades off our desire for a sparse solution with our wish for a low approximation error).

There are many regularizers that promote sparsity. For example, the ℓ_p -norms, $0 \leq p \leq 1$ are well-known to promote sparsity:

$$f(\mathbf{x}) = \|\mathbf{x}\|_p^p = \sum_i |x_i|^p \quad (14)$$

We can begin to see why the ℓ_p -norms, $0 \leq p \leq 1$, promote sparsity by visualizing the shape of ℓ_p -balls in two dimensions. In fig. 1 the feasible set of solutions for a hypothetical problem is indicated by a dashed line. The minimum ℓ_p -norm solution is found by expanding the ℓ_p -ball until it intersects the solution space. As can be seen for $p \leq 1$ the ℓ_p -ball intersects the solution space along one of the coordinate axes, leading to a solution with only one non-zero component. Notice that when $p > 1$ the solution contains two non-zero components. The solution for $p = 2$ corresponds to the minimum energy solution (which is calculated using the method of frames). Geometrically the ℓ_p -balls, $0 \leq p \leq 1$, are sharply pointed and aligned with the coordinate axes, which tends to induce sparse solutions.

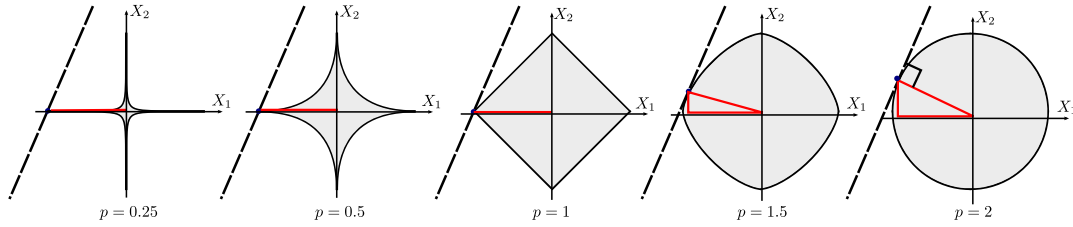


Figure 1: Illustration of the shape of ℓ_p -balls in 2-dimensions. The dashed line represents the set of feasible solutions for a hypothetical problem. Note that for $p > 1$ the minimum ℓ_p -norm solution contains 2 non-zero components, whereas for $p \leq 1$, the solution contains only 1 non-zero component. Also note that $p = 2$ is the minimum energy solution (it is the vector normal to the solution space).

Other sparsity measures are also possible, for example, the Shannon and Rényi entropies are known to act as good sparsity measures² [24, 25].

5.1. Basis Pursuit

The fact that ℓ_1 minimization often leads to sparse solutions has been known for sometime [26]. In the signal processing literature minimization of (12) with an ℓ_1 -norm regularization term is known as basis pursuit (BP) when $T = 0$ and basis pursuit denoising (BPDN) when $T \neq 0$ [27]. In the statistics literature a very similar formulation was presented under the name least absolute shrinkage and selection operator (LASSO) [28].

Using the ℓ_1 -norm is attractive since *i*) it promotes sparsity, and *ii*) it is convex (and thus this problem can be tackled using the large body of techniques developed for convex optimization [29]).

In [27] the BP problem was solved using linear programming (LP), however, as pointed out in [30], LP cannot be used with complex coefficients. In this case a second-order cone program (SOCP) may be used instead.

In [30] it was noted that we can downsample a sparse signal using random projections (using results from compressed sensing (CS) theory [31]). This strategy was used in [30] to downsample the input signal before applying a SOCP. This significantly reduces the problem size and hence the computational cost (which is very interesting to note for audio applications).

In the following sections we discuss algorithms for the solution to the unconstrained problem (13).

5.2. Iteratively Re-weighted Least Squares

Iteratively re-weighted least squares (IRLS) is an algorithm that can be used to solve the sparse approximation problem with both convex and non-convex regularization terms. The premise of IRLS stems from the following fact: if we define a diagonal weight matrix as:

$$W_p = \text{diag}(|x_i|^{p-2}) \quad (15)$$

then we can write the ℓ_p -norm of \mathbf{x} in quadratic form as follows:

$$\|\mathbf{x}\|_p^p = \mathbf{x}^H W_p \mathbf{x} = \sum_i x_i^2 |x_i|^{p-2} = \sum_i |x_i|^p \quad (16)$$

This allows us to write (13) as:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - \Phi \mathbf{x}\|_2^2 + \lambda \mathbf{x}^H W_p \mathbf{x} \quad (17)$$

²In fact, the Rényi entropies can be interpreted as logarithmic versions of the ℓ_p -norms.

The least squares solution to this equation is:

$$\hat{\mathbf{x}} = (\Phi^H \Phi + 2\lambda W_p)^{-1} \Phi^H \mathbf{y} \quad (18)$$

However, since $W_p = \text{diag}(|x_i|^{p-2})$ is a function of \mathbf{x} , we must solve this equation in an iterative fashion. The pseudocode in algorithm 2 demonstrates the basic IRLS algorithm. To avoid division by zero, we initialize \mathbf{x} with all ones. In practice many of the coefficients of \mathbf{x} will shrink, but never reach zero. A variation on this algorithm could include the identification of an *active set* of coefficients from \mathbf{x} . Small coefficients from \mathbf{x} (and the associated columns from Φ) could then be pruned from the active set.

Algorithm 2 IRLS

- 1: **init:** $n = 0, \mathbf{x}^{(n)} = \mathbf{1}$
 - 2: **repeat**
 - 3: $W_p^{(n)} = \text{diag}(|x_i^{(n)}|^{p-2})$
 - 4: $\mathbf{x}^{(n+1)} = (\Phi^H \Phi + 2\lambda W_p^{(n)})^{-1} \Phi^H \mathbf{y}$
 - 5: $n = n + 1$
 - 6: **until** stopping condition
-

In [32] a slightly different procedure is described, whereby a solution is sought in the noiseless case. This algorithm has the same form as algorithm 2, except that at each iteration the updates proceed according to

$$\mathbf{x}^{(n+1)} = (W^{(n)})^{-1} \Phi^H (\Phi (W^{(n)})^{-1} \Phi^H)^{-1} \mathbf{y} \quad (19)$$

This variant of IRLS is referred to as the focal underdetermined system solver (FOCUSS) in the literature [33]. A detailed examination of IRLS and its convergence properties is provided in [34].

There are several points that should be noted regarding the IRLS algorithm. Firstly, the algorithm is sensitive to the initialization point when $p < 1$, which means a local minimum could be returned. In many applications we may be able to find a suitable initialization point using other algorithms (e.g., we could use the pseudo-inverse as an initialization point). Second, the algorithm requires a matrix inversion. It can be practical to perform the matrix inversion using Cholesky or QR factorization for small dictionaries. However, when working with audio, these factorizations are usually not practical due to their computational complexity, and because we often can't explicitly store the dictionary in memory. In this case we can perform the matrix inversion using conjugate gradient descent with appropriate preconditioning as suggested in [35]. As noted in [36] when Φ is an orthonormal basis the matrix inverse is trivial. Furthermore, if Φ is a union of orthonormal bases, we can invert one basis at a time in an iterative fashion using block coordinate relaxation (BCR) [37].

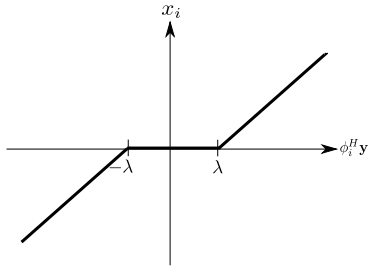


Figure 2: The shrinkage curve $x_i = \psi(\phi_i^H \mathbf{y})$ for the ℓ_1 -norm function.

5.3. Iterative Shrinkage

In [38] a method known as shrinkage was introduced to solve (13) for the case when Φ is an orthonormal basis. To understand shrinkage, we start by re-writing the objective function from (13) as:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2} \|\Phi(\Phi^{-1} \mathbf{y} - \mathbf{x})\|_2^2 + \lambda f(\mathbf{x}) \quad (20)$$

when Φ is an orthonormal basis, this simplifies to³:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2} \|\Phi^H \mathbf{y} - \mathbf{x}\|_2^2 + \lambda f(\mathbf{x}) \quad (21)$$

$$= \arg \min_{\mathbf{x}} \sum_i \frac{1}{2} (\phi_i^H \mathbf{y} - x_i)^2 + \lambda f(x_i) \quad (22)$$

In this form the joint optimization problem has been factored into a sum of scalar optimization problems which can be solved individually.

The solution to the i^{th} scalar optimization problem is given by:

$$x_i + \lambda f'(x_i) = \phi_i^H \mathbf{y} \quad (23)$$

Solving for x_i we get:

$$x_i = \psi(\phi_i^H \mathbf{y}) \quad (24)$$

where

$$\psi^{-1}(u) = u + \lambda f'(u) \quad (25)$$

When f is the ℓ_1 -norm we find that $\psi^{-1}(u) = u + \lambda \text{sgn}(u)$, which leads to:

$$\psi(u) = \text{sgn}(u) \max(0, |u| - \lambda) \quad (26)$$

The curve of the ℓ_1 shrinkage function, ψ , is graphed in figure 2. Filtering the transform coefficients according to this graph is known as shrinkage or soft thresholding. It was shown in [39] that shrinkage can be performed with complex coefficients by simply shrinking the modulus of $\phi_i^H \mathbf{y}$ and leaving the argument unchanged. We also note that different regularization terms will lead to different shrinkage curves⁴. For example, in the limit as $p \rightarrow 0$ we obtain a variant known as hard thresholding [40].

We now discuss how to perform shrinkage with an overcomplete dictionary. In this case an iterative shrinkage (IS) algorithm is required, whereby a series of simple shrinkage operations are performed until convergence [41, 42, 43, 44, 40]. Here we outline the approach discussed in [43].

³by Parseval's theorem $\|\Phi \mathbf{z}\| = \|\mathbf{z}\|$, and the fact that $\Phi^{-1} = \Phi^H$

⁴Not all shrinkage functions can be calculated analytically. In such a case a look-up table can be used.

In this approach we again decouple the optimization problem and solve a series of 1-D problems. Assume we are given the transform coefficients at the n^{th} iteration, $\mathbf{x}^{(n)}$. Now assume all entries in $\mathbf{x}^{(n)}$ are fixed, except for the i^{th} entry, which we wish to refine. We can write the objective function as:

$$\arg \min_w \frac{1}{2} \|\mathbf{y} - (\Phi \mathbf{x}^{(n)} - \phi_i x_i^{(n)} + \phi_i w)\|_2^2 + \lambda f(w) \quad (27)$$

In essence this removes the contribution of the i^{th} atom from the model, and allows us to replace it with a new estimate, w . Taking the derivative with respect to w and setting the result to zero we get (assuming unit norm atoms):

$$w + \lambda f'(w) = x_i^{(n)} + \phi_i^H (\mathbf{y} - \Phi \mathbf{x}^{(n)}) \quad (28)$$

which is in the same form as (23), and so can be solved using a shrinkage operator. The pseudocode for iterative shrinkage (IS) listed in algorithm 3:

Algorithm 3 Iterative Shrinkage (IS)

```

1: init:  $n = 0, \mathbf{x}^{(n)} = \mathbf{0}$ 
2: repeat
3:    $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)}$ 
4:   for  $i = 0$  to  $N - 1$  do
5:      $x_i^{(n+1)} = \psi(x_i^{(n)} + \phi_i^H (\mathbf{y} - \Phi \mathbf{x}^{(n+1)}))$ 
6:   end for
7:    $n = n + 1$ 
8: until stopping condition

```

In [43], empirical results are presented comparing IS to IRLS. Although IRLS has much faster convergence, it also requires a matrix inversion, which can be prohibitive for large dictionaries. Iterative shrinkage on the other hand, converges more slowly, but it is computationally much simpler (and does not require a matrix inversion), so it can easily be used with large overcomplete dictionaries.

It is important to note that IS tends to underestimate the magnitude of the synthesis coefficients. We can correct for this bias after running IS by taking the orthogonal projection of \mathbf{y} onto the support identified by the algorithm. In practice there may be some coefficients that are very small (but non-zero). These coefficients can be eliminated by hard thresholding prior to the debiasing step.

The particular version of shrinkage in algorithm 3 is somewhat slow because it requires each x_i to be updated sequentially in the inner loop. In [43] a simple modification of this algorithm was introduced that allows all of the coefficients in \mathbf{x} to be updated in parallel. This leads to a significant speed up in the algorithm. We note that the IS algorithm developed in [42] also uses a parallel update.

There have also been several recent papers introducing fast IS techniques [45, 46]. These algorithms use information from the two previous iterates to update the current solution, and can be up to an order of magnitude faster. These fast shrinkage algorithms would likely be quite useful for audio applications.

5.4. Bayesian Methods

In a probabilistic setting we assume the signal is constructed according to:

$$\mathbf{y} = \Phi \mathbf{x} + \varepsilon \quad (29)$$

where \mathbf{x} and ϵ are random variables representing the signal and noise, respectively⁵. We will assume that ϵ is white Gaussian noise with covariance matrix $\Sigma = \sigma^2 I$ in order to simplify calculations (however more general noise models are certainly possible).

The negative log-likelihood of \mathbf{x} is given by

$$\mathcal{L}(\mathbf{x}) = -\log p(\mathbf{y}|\mathbf{x}) = \frac{1}{2\sigma^2} \|\mathbf{y} - \Phi\mathbf{x}\|_2^2 + C \quad (30)$$

where C is a constant independent of \mathbf{x} . As discussed in §2 when Φ is overcomplete there are multiple values of \mathbf{x} such that $\mathbf{y} = \Phi\mathbf{x}$, which means the maximum likelihood solution is ill-defined.

In order to find sparse solutions we can instead find the maximum *a priori* (MAP) estimate which incorporates a prior on the transform coefficients. The MAP estimate is found by application of Baye's rule:

$$\hat{\mathbf{x}}_{MAP} = \arg \max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) = \arg \max_{\mathbf{x}} \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \quad (31)$$

It is usually mathematically simpler to minimize the negative log-likelihood which is:

$$\hat{\mathbf{x}}_{MAP} = \arg \min_{\mathbf{x}} -\log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{x}) + \log p(\mathbf{y}) \quad (32)$$

Let us denote the negative log-probability of \mathbf{x} as $f(\mathbf{x}) = -\log p(\mathbf{x})$. Then (32) becomes

$$\hat{\mathbf{x}}_{MAP} = \arg \min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y} - \Phi\mathbf{x}\|_2^2 + f(\mathbf{x}) + C \quad (33)$$

where C is a constant independent of \mathbf{x} . Notice that (33) is essentially the same problem as (13). It is interesting to note that when $p(\mathbf{x})$ is an independent and identically distributed (i.i.d.) Laplacian prior:

$$p(\mathbf{x}) = \prod_i p(x_i) = \prod_i \frac{\lambda}{2} \exp(-\lambda|x_i|) \quad (34)$$

then

$$f(\mathbf{x}) = -\log p(\mathbf{x}) = \lambda \|\mathbf{x}\|_1 + C \quad (35)$$

In other words, MAP estimation with a Laplacian prior is equivalent to ℓ_1 regularization. The Laplacian distribution is sharply peaked at zero with heavy tails as illustrated in fig. 3. This density thus encourages many small coefficients, yet does not place a heavy penalty on large coefficients, which tends to promote sparse solutions. This new viewpoint helps to illustrate why the ℓ_1 -norm acts as a good sparsity measure. It should be noted that this is just one possible interpretation, and that other interpretations are certainly possible as suggested in the recent paper [48].

Since we are free to investigate priors other than the Laplacian, and also because we can use alternative noise models, the Bayesian approach is quite flexible. For example, in [49], a piecewise continuous prior was used, which is even more peaked around zero than the Laplacian prior. In [50], a design methodology is discussed outlining some of the necessary conditions for a prior to be sparsity inducing.

It should be noted that in the above formulation the synthesis coefficients were assumed to be i.i.d.. In reality this may not be a good assumption, since we expect some structure in the synthesis coefficients (for example chains of coefficients that form partials). In fact, all of the algorithms discussed up to this point ignore this important factor. We discuss methods for sparse and structured decompositions in §6.

⁵In some cases Φ can also be considered a random matrix, for example, if we wish to perform dictionary learning [47].

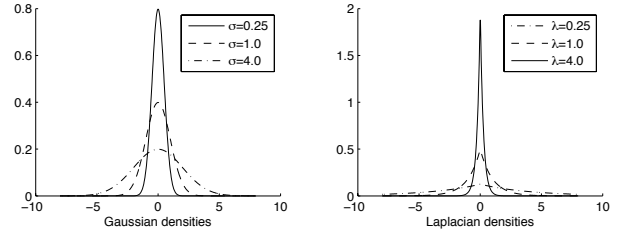


Figure 3: Illustration of Gaussian densities (left) and Laplacian densities (right).

5.4.1. Bayesian Variants and Extensions

There are several variations that can be applied within the Bayesian framework. For example, in [51] [52] [53] each transform coefficient is modeled by a zero-mean Gaussian prior. A hyper-prior is then placed on the variance, in order to express the belief that, for the majority of coefficients, the variance should be near zero [53]. As illustrated in fig. 3, as the variance of a Gaussian tends to zero, the distribution becomes infinitely peaked at zero. An expectation maximization (EM) procedure can be used to find MAP solutions for these models. A conceptually similar approach is discussed in [54].

Another variant known as sparse linear regression was introduced in [55] and [56]. In this formulation the atomic model is augmented to include a binary indicator variable $\gamma_i \in \{0, 1\}$:

$$\mathbf{y} = \sum_{i=0}^{N-1} \gamma_i x_i \phi_i + \epsilon \quad (36)$$

The vector $\gamma = [\gamma_0, \dots, \gamma_{N-1}]^T$ indicates the presence or absence of each coefficient in the model. If we could somehow determine a sparse indicator vector, then we could find the optimal transform coefficients via orthogonal projection. This problem setup is known in the literature as Bayesian variable selection [57].

Using the indicator variables we can form a mixed prior for each transform coefficient:

$$p(x_i|\gamma_i, \sigma_i) = (1 - \gamma_i)\delta(x_i) + \gamma_i\mathcal{N}(x_i|0, \sigma_i^2) \quad (37)$$

where $\mathcal{N}(\cdot|0, \sigma^2)$ indicates a zero-mean Gaussian density with variance σ^2 , and $\delta(\cdot)$ is a dirac distribution. This *spike + slab* distribution enforces sparsity conditionally on γ . We then seek a MAP solution of the form

$$(\hat{\mathbf{x}}, \hat{\gamma}) = \arg \max_{\mathbf{x}, \gamma} p(\mathbf{x}, \gamma|\mathbf{y}) \quad (38)$$

where the density $p(\mathbf{x}, \gamma|\mathbf{y})$ can be found by marginalizing the complete posterior. The type of prior we place on γ , strongly effects the type of solutions that will be found. For example, an independent Bernoulli prior could be used if we don't expect any dependencies between coefficients. In [58] the indicator variables are given time-frequency dependencies by modeling the joint distribution of γ as a Markov chain. In general the density $p(\mathbf{x}, \gamma|\mathbf{y})$ cannot be found analytically. In this case, a solution can be found using Monte Carlo inference (e.g., Gibbs sampling).

6. STRUCTURED APPROXIMATIONS

The majority of techniques discussed to this point simply aim at recovering a sparse solution to (1). In addition to sparsity, we of-

ten have other *a priori* information regarding the content of our signals. In audio signals we expect strong time-frequency dependencies between different transform coefficients. For example, we expect some time-continuity between active coefficients in tonal sounds, and some frequency-continuity in transient sounds. Furthermore, we expect spatial dependencies in multichannel recordings [59].

There are several ways of structuring the information in a decomposition. We can impose structure directly using the atoms themselves (for example, by using harmonic atoms). Likewise, we can impose structure by preferentially selecting coherent atoms in the decomposition (for example, by only selecting overlapping chains of atoms). Finally, we can impose structure as a post-processing step, by clustering atoms according to their similarity. All of these approaches have been used in the literature. We briefly review the most prominent techniques.

Sparse linear regression, which was addressed in the previous section, uses binary indicator variables to indicate the absence or presence of each transform coefficient in the model. The type of joint prior placed on the set of indicator variables can be used to model dependencies between the transform coefficients, as was done in [55][58].

An algorithm known as molecular matching pursuit (MMP) is described in [60]. MMP is an iterative greedy algorithm that estimates and subtracts a tonal or transient molecule from the residual at each stage. Tonal molecules are defined as overlapping chains of MDCT atoms with the same frequency (± 1 bin). Transient molecules are sets of wavelet coefficients forming a connected tree. A similar algorithm is described in [13] which uses a multi-scale Gabor dictionary. These algorithms segment the signal into transient and tonal objects, which could be useful for selective processing and filtering of audio signals. There are strong similarities between MMP and partial tracking algorithms [61, 62], especially when an overlap-add (OLA) synthesis model is used. However, the advantage of MMP and its variants over traditional partial tracking techniques is that multi-scale features can be captured by the algorithm, since larger, more general dictionaries can be used.

An algorithm known as harmonic matching pursuit (HMP) was suggested in [23]. This algorithm uses a dictionary of harmonic atoms⁶. At each stage in the pursuit the signal is projected onto a harmonic subspace. In order to manage the complexity of the algorithm the search for best harmonic atom is limited to a subset of the dictionary using a weak approximate MP. The same authors also developed stereo atoms for two channel recordings [59].

The authors in [63] also discussed an approach inspired by both harmonic matching pursuit and molecular matching pursuit. Their mid-level representation was shown to be useful for several tasks including solo and polyphonic instrument recognition.

In [64] a post-processing technique known as agglomerative clustering (AC) was introduced to impose structure on the synthesis coefficients. AC works by traversing an adjacency matrix, which measures the similarity between atoms. If two atoms are present in the decomposition, and significantly close in the adjacency matrix, then they are grouped together. This process is then repeated to form large clusters of coherent atoms.

In [65] a technique using iterative shrinkage (see §5.3) was developed to find sparse and structured decompositions when used

⁶a harmonic atom is defined as a weighted sum of Gabor atoms with integer frequency ratios.

Package	MP	OMP	IRLS	IS	BP	Languages	URL
SparseLab	✓	✓	✓	✓	✓	Matlab	[66]
Sparsify	✓	✓	-	-	-	Matlab	[67]
MPTK	✓	-	-	-	-	C++ ⁷	[68]
GabLab	✓	✓	✓	✓	-	Matlab	[69]

Table 1: Software packages for sparse approximation.

with time-frequency dictionaries. This technique relies on a mixed-norm regularization term which tends to induce structure in the decomposition (for details see [65] and the references therein).

7. SOFTWARE TOOLS

Table 1 lists several software packages for sparse approximation that are freely available. This list is by no means exhaustive, but it does include some of the more well-known choices that are available.

In practice not all packages are well-suited for audio analysis. For example, many of the solvers in SparseLab require explicit access to the columns or rows of the dictionary. This is problematic for audio analysis, since we often work with huge amounts of data and dictionaries that aren't explicitly stored.

In practice MPTK is very fast, and contains many optimizations that make it suitable for use with audio and very large shift-invariant dictionaries.

The GabLab software [69] (which has been released in conjunction with this paper) was written with audio analysis in mind. GabLab comes bundled with functions for creating fast Gabor dictionaries, and unions of Gabor frames⁸. The computation of inner products in GabLab is performed using the fast Fourier transform. Furthermore, all of the algorithms in GabLab work with complex atoms.

In the following sections we compare the four main algorithms discussed in this paper (MP, OMP, IS and IRLS) according to several different criteria. We then discuss several audio applications that could benefit from sparse atomic modeling. All of the numerical experiments were performed using GabLab.

8. COMPUTATIONAL COMPLEXITY

A detailed analysis of the computational complexity of MP and OMP for general and fast local dictionaries can be found in [10]. We note that MP and IS are each dominated by the calculation of inner products, and thus have a similar computational complexity. However, when local dictionaries are used the inner product update can be performed faster in the MP algorithm (since fewer inner products need to be calculated for each iteration).

OMP and IRLS are both dominated by the calculation and inversion of the (partial) Gram matrix at each iteration. In GabLab this matrix inversion is performed using conjugate gradient descent.

⁷Matlab bindings are bundled with the MPTK distribution.

⁸A Gabor frame can be viewed as a generalization of an oversampled STFT matrix.

The difference in speed between these algorithms depends to a large degree on the number of iterations required for convergence. In the following section we provide empirical results which illustrate how many iterations of each algorithm are required in specific test cases.

9. COMPARISON

In this section we compare the performance of MP, OMP, IS ($p=1$), and the IRLS ($p=1$) algorithm using some simple synthetic test signals. In the following tests we used a 3-scale complex Gabor dictionary constructed from Hann windows of length 2048, 512, and 64 samples with 50% overlap. The sampling rate used was 44.1kHz.

9.1. Example: a compressible signal

For the first test we used a quadratic chirp swept between 100 Hz and 15 kHz. As can be seen from the spectrogram in fig. 4, this signal is compressible in the Fourier domain, i.e., many of the coefficients are small. Furthermore, since the bandwidth of the chirp evolves over time, a multi-scale dictionary (such as the one proposed above), should possess the capability to model both the narrowband and wideband parts of the chirp respectively.

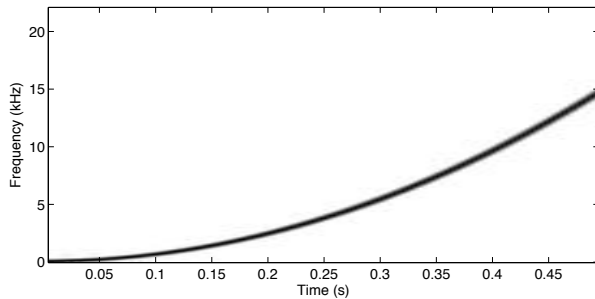


Figure 4: Spectrogram of quadratic chirp test signal. Spectrogram parameters: Hann window of length 512, with 50% overlap.

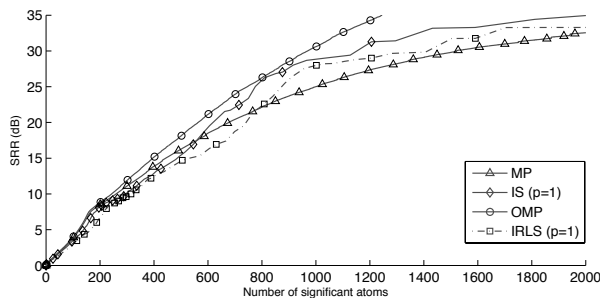


Figure 5: SRR vs. Number of significant atoms for quadratic chirp decomposition.

Figure 5 shows the signal to residual ratio (SRR) vs. the number of significant atoms in the decomposition for each algorithm. An atom was deemed significant if its magnitude exceeded 10^{-4} . For IS and IRLS the data points were generated by running the algorithms until convergence for different values of λ . It should be noted that this process can be accelerated significantly using

SRR	10 dB			20 dB			30 dB		
	iterations	# of atoms	cpu time (s)	iterations	# of atoms	cpu time (s)	iterations	# of atoms	cpu time (s)
Algorithm									
MP	131	260	8.0	354	678	21.4	844	1521	51.1
IS	92	292	6.6	151	618	10.6	230	1127	16.5
OMP	122	242	116.0	282	560	381.2	484	960	1224.6
IRLS	73	315	89.0	69	733	127.2	67	1406	206.5

Table 2: Summary of quadratic chirp decomposition results.

‘warm starts’, i.e., initializing the next run of the algorithm using the previous value at convergence. This works in practice because a small change in λ usually causes a small change in the solution. It should also be noted that after IS and IRLS converged the coefficient vector was thresholded (with threshold 10^{-4}) and debiased as explained in section §5.3.

Examining the curves in fig. 5, we see that OMP provides the best SRR vs. number of atoms. MP, IS, and IRLS perform similarly, although beyond 800 atoms, IS and IRLS both maintain a higher SRR than MP. Table 2 summarizes the data in this graph and adds two new pieces of information: i) the number of iterations required for convergence and, ii) the amount of CPU time used by each of the algorithms (for reference, all of the algorithms were run in MATLAB[®] on the same 2.6 GHz dual core Mac Pro). Of course, the speed of these algorithm is implementation dependent. However, combined with the number of iterations required for convergence these numbers do reveal interesting differences between the various algorithms.

As described in §8, IS and MP have approximately the same complexity per iteration, however, as seen in table 2, IS requires fewer iterations to converge than MP, and hence uses less CPU time. The difference is more dramatic for high SRRs and, although not shown in table 2, for very low SRRs MP is indeed faster.

9.2. Example: a sparse signal plus noise

For this example, we generated a random sparse signal using the 3-scale Gabor dictionary introduced in the previous section. This signal was generated by first drawing 500 indices from a uniform distribution to make up the support vector. The real and imaginary coefficients were then drawn from a normal distribution with unit mean and variance 0.1. Conjugate atoms were also added to make the signal real. The test signal was 0.5s in duration and contained 984 non-zero coefficients⁹. We then added white Gaussian noise to the signal so that the SNR was 5dB and compared the performance of MP, OMP, IS and IRLS at denoising the signal.

Figure 6 displays the output SNR vs. number of atoms for each of the algorithms. Near the true sparsity level (984 atoms), OMP offers the best reconstruction in terms of output SNR. MP has its peak located in a similar location, although the SNR is lower¹⁰. Both IS and IRLS require more atoms to reach their peak SNR,

⁹There are slightly less than 1000 non-zero coefficients because repeated coefficients were discarded, and some coefficients were DC atoms (so no conjugate was added).

¹⁰Stopping the algorithm here and running backprojection would probably result in a better performance.

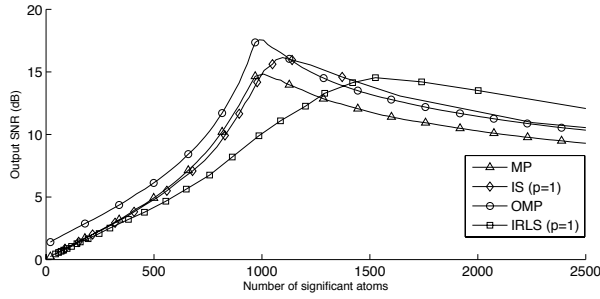


Figure 6: Results for de-noising a random sparse signal corrupted by white Gaussian noise (input SNR = 5dB).

although IS achieves its peak with fewer significant atoms than IRLS.

In order to compare how well the true support was recovered we also measure the Type I and II errors which are defined as follows. If we let Δ represent the true support and $\hat{\Delta}$ represent the estimated support:

$$\text{Type I error} \triangleq 1 - \frac{|\Delta \cap \hat{\Delta}|}{|\hat{\Delta}|} \quad (39)$$

$$\text{Type II error} \triangleq 1 - \frac{|\Delta \cap \hat{\Delta}|}{|\Delta|} \quad (40)$$

Figure 7 illustrates the Type I and Type II errors vs. the number of atoms for each of the algorithms. MP, OMP, and IS all have very low errors near 984 atoms (the true sparsity level), which suggests that these algorithms do a good job recovering the correct support. IRLS on the other hand, has a harder time recovering the true support. It is interesting to note the differences between IS and IRLS since both algorithms attempt to minimize the same cost function. We must remember however, that this cost function is not *strictly* convex, which means there could be multiple solutions with the same cost.

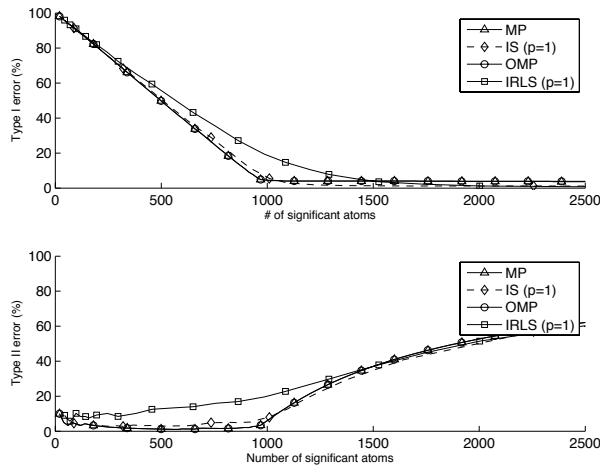


Figure 7: Error in support estimation. Top: Type I error. Bottom: Type II error.

10. AUDIO APPLICATIONS

In this section we highlight some audio applications that can benefit from sparse atomic modeling.

10.1. Multilayer Expansions

Multilayer expansions are often very useful for audio processing and effects. For example, the *tonal + transient + noise* expansion segments the signal into important perceptual units.

To find a multilayered expansion we start by defining the dictionary as $\Phi = \cup_{i=1}^I \Phi_i$, where each Φ_i is a frame adapted to a certain signal feature. For example, for $I = 2$, we might take Φ_1 to be a Gabor frame with a short duration window and Φ_2 to be a Gabor frame with a long duration window. These frames are well suited for the analysis of transient and tonal structures, respectively.

Now, if we solve the system $\mathbf{y} = \Phi \mathbf{x} = [\Phi_1 \Phi_2] [\mathbf{x}_1 \mathbf{x}_2]^T$ with a sparsity constraint, it follows that the transient components will be encapsulated by \mathbf{x}_1 and tonal components will be encapsulated by \mathbf{x}_2 ¹¹.

For example, fig. 8 shows the multilayer analysis of a 5s long glockenspiel excerpt, which was chosen because it has rather distinct tonal and transient parts. The analysis was performed with a 2-scale Gabor dictionary with Hann windows of length 2048 and 32 samples with 50% overlap. The sampling rate used was 44.1kHz. The particular analysis shown was performed using IS, however, all of the algorithms discussed lead to fairly similar results. The interested reader can listen to the multilayer expansions found using MP, IS, and IRLS on the companion website [69].

10.2. Denoising

As shown in the example presented in §9.2, prior knowledge of sparsity or compressibility is often useful for signal denoising. Further, as discussed in §5.4, regularization with a sparse prior can be interpreted as a MAP estimate in certain situations. An additional denoising example using the glockenspiel excerpt from the previous section can be found on the companion website [69].

10.3. Time-Frequency Modification

In this paper we have primarily focused on the use of Gabor frames and sparsity of the synthesis coefficients. This point-of-view is useful for time-frequency modifications, since the synthesis coefficients of a Gabor frame can be used to control the time-frequency content of the signal. For example, on our companion website [69] we include an example of a major-to-minor transposition of an acoustic guitar chord. This effect was achieved using the following steps:

1. A tonal + transient expansion was performed as described in §10.1.
2. The tonal atoms were then classified based on whether or not they belonged to the major third note in the chord (this requires an multiple f_0 estimation).
3. The major third atoms were then synthesized and flattened by 100 cents to produce a minor third note.
4. The original signal was then re-synthesized without the major third atoms and added to the minor third signal to produce a minor chord.

¹¹Provided Φ_1 and Φ_2 are incoherent with one another.

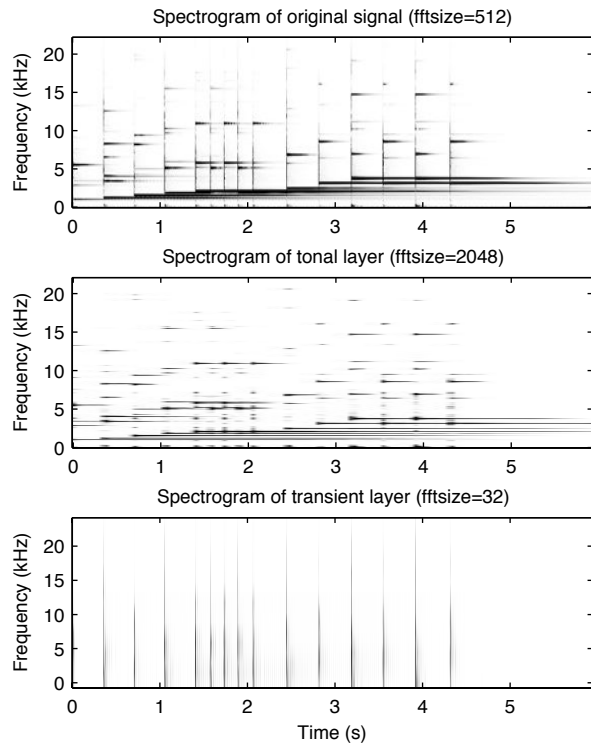


Figure 8: Multilayered expansion of a glockenspiel excerpt. Top: original spectrogram. Middle: tonal layer. Bottom: transient layer. In this example the total SRR was 32 dB and 2% of the synthesis coefficients were non-zero.

As can be heard on the website [69] the result is relatively convincing despite the naïvety of this approach. We also provide a similar example of time-stretching with transient preservation on the website [69].

We can also selectively process atoms by type. For example short duration atoms could be attenuated or amplified to smooth or accentuate an attack.

10.4. Granular Synthesis

As discussed in [70] the sparse synthesis model can be used to achieve many standard granular effects. For example, we can:

1. Apply time and frequency jitter to the atom parameters.
2. Change the atom envelopes.
3. Change the atom durations (*bleed*).
4. Change the density of atoms per unit time.

We provide several audio examples of these effects on the companion website [69].

10.5. Inverse-problems

In some cases we may not be able to directly observe the true signal. For example, we may be forced to work with limited data due to hardware requirements or assumptions regarding stationarity. In other cases we may only have access to a noisy or reverberant signal. Likewise, the signal might be downsampled, have small gaps,

or be corrupted with clicks. We can often describe these types of degradations as:

$$\mathbf{z} = \Psi \mathbf{y} + \varepsilon \quad (41)$$

where \mathbf{z} is the observed signal, Ψ is a (known) linear degradation operator, \mathbf{y} is the true signal, and ε is additive noise. If \mathbf{y} has a sparse representation $\mathbf{y} = \Phi \mathbf{x}$ then we can re-write (41) as

$$\mathbf{z} = \mathbf{D} \mathbf{x} + \varepsilon \quad (42)$$

where $\mathbf{D} = \Psi \Phi$. Armed with the knowledge that \mathbf{x} is sparse, we can attempt to estimate $\hat{\mathbf{x}}$ using the dictionary \mathbf{D} and any of the techniques discussed in this paper. We can then generate an estimate of the true signal as $\hat{\mathbf{y}} = \Phi \hat{\mathbf{x}}$. Under certain conditions regarding \mathbf{D} and the sparsity of \mathbf{x} , it is possible to exactly recover \mathbf{y} [71]. This premise was recently applied in [72] for audio restoration.

It has also been shown that when Ψ is a random matrix, \mathbf{y} can be recovered (with high probability) if the number of rows in Ψ is large enough. This is the basis of compressed sensing (CS) [31].

11. CONCLUSION

In this paper we reviewed sparse atomic models for audio and discussed several algorithms that can be used to estimate the model parameters. This included an exploration of greedy, relaxed, and Bayesian approaches to the sparse approximation problem, as well as a brief look at structured approximations. Further, we provided a few numerical comparisons that serve to illustrate some of the practical differences between the algorithms discussed. Lastly we included a discussion of several interesting audio applications that can benefit from sparse atomic modeling. We remind the reader that many of the examples in this paper along with sound files and MATLAB[®] code can be found online [69].

We are currently working on MATLAB[®] implementations of local OMP [10], and fast IS [46], which will be added to a future release of GabLab. We also plan to implement several structured decomposition techniques and to expand upon the comparisons performed in this paper.

12. REFERENCES

- [1] M. Elad, P. Milanfar, and R. Rubinstein, “Analysis versus synthesis in signal priors,” *Inverse Problems*, vol. 23, pp. 947–968, 2007.
- [2] S. Nam, M. Davies, M. Elad, and R. Gribonval, “Cosparsity analysis modeling-uniqueness and algorithms,” *Proc. Int. Conf. Acoust. Speech Sig. Proc.*, 2011.
- [3] I. Daubechies, A. Grossmann, and Y. Meyer, “Painless non-orthogonal expansions,” *J. Math. Phys.*, vol. 27, no. 5, pp. 1271–1283, 1986.
- [4] J. Kovacevic and A. Chebira, “Life Beyond Bases: The Advent of Frames (Part I),” *IEEE Sig. Proc. Mag.*, vol. 24, no. 4, pp. 86–104, July 2007.
- [5] M. Goodwin, “Matching pursuit with damped sinusoids,” in *Proc. Int. Conf. Acoust. Speech Signal Process*, 1997, vol. 3, pp. 2037–2040.
- [6] G. Davis, S. Mallat, and Z. Zhang, “Adaptive time-frequency decompositions with matching pursuit,” in *Proc. of SPIE*, 1994, vol. 2242, pp. 402–413.

- [7] J.A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.
- [8] J.A. Tropp, "Just relax: Convex programming methods for identifying sparse signals in noise," *IEEE Trans. on Inf. Theory*, vol. 52, no. 3, pp. 1030–1051, 2006.
- [9] S.G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. on Sig. Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.
- [10] B. Mailhé, R. Gribonval, P. Vanderghenst, and F. Bimbot, "Fast orthogonal sparse approximation algorithms over local dictionaries," *Signal Processing*, 2011.
- [11] R. Heusdens, R. Vafin, and W.B. Kleijn, "Sinusoidal modeling using psychoacoustic-adaptive matching pursuits," *Signal Processing Letters, IEEE*, vol. 9, no. 8, pp. 262–265, 2002.
- [12] R. Gribonval, E. Bacry, S. Mallat, P. Depalle, and X. Rodet, "Analysis of sound signals with high resolution matching pursuit," in *Proc. of the IEEE Int. Sym. on Time-Freq. and Time-Scale Analysis*, 1996, pp. 125–128.
- [13] P. Leveau and L. Daudet, "Multi-resolution partial tracking with modified matching pursuit," in *Proc. Eur. Signal Process. Conf.*, 2006.
- [14] M.G. Christensen and S.H. Jensen, "The cyclic matching pursuit and its application to audio modeling and coding," in *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, IEEE, 2007, pp. 550–554.
- [15] Y.C. Pati, R. Rezaifar, and P.S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," *Asilomar Conf. on Sig., Sys. and Comp.*, vol. 1, pp. 40–44, 1993.
- [16] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *Signal Processing Letters, IEEE*, vol. 9, no. 4, pp. 137–140, 2002.
- [17] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean-Luc Starck, "Sparse Solution of Underdetermined Linear Equations by Stagewise Orthogonal Matching Pursuit," *Tech. Rep.*, Stanford, 2006.
- [18] T. Blumensath and M.E. Davies, "Gradient pursuits," *IEEE Trans. on Sig. Proc.*, vol. 56, no. 6, pp. 2370–2382, 2008.
- [19] M. Goodwin, *Handbook of Speech Processing*, chapter 12, pp. 229–258, Springer-Verlag, Berlin, 2008.
- [20] R. Gribonval and M. Nielsen, "Approximate weak greedy algorithms," *Adv. in Comp. Math.*, vol. 14, no. 4, pp. 361–378, 2001.
- [21] V.N. Temlyakov, "Weak greedy algorithms," *Adv. in Comp. Math.*, vol. 12, no. 2-3, pp. 213–227, 2000.
- [22] F. Bergeaud and S. Mallat, "Matching pursuit: Adaptive representations of images and sounds," *Comp. and Applied Math.*, vol. 15, pp. 97–110, 1996.
- [23] R. Gribonval and E. Bacry, "Harmonic decomposition of audio signals with matching pursuit," *IEEE Trans. on Sig. Proc.*, vol. 51, no. 1, pp. 101–111, 2003.
- [24] R.R. Coifman and M.V. Wickerhauser, "Entropy-based algorithms for best basis selection," *IEEE Trans. on Inf. Theory*, vol. 38, no. 2, pp. 713–718, 1992.
- [25] F. Jaillet and B. Torrèsani, "Time-Frequency Jigsaw Puzzle: adaptive multiwindow and multilayered Gabor expansions," *Int. J. Wavelets, Multires. and Inf. Proc.*, vol. 5, no. 2, pp. 293–315, 2007.
- [26] J.F. Claerbout and F. Muir, "Robust modeling with erratic data," *Geophysics*, vol. 38, pp. 826, 1973.
- [27] S.S. Chen, D.L. Donoho, and M.A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.
- [28] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *J. Royal Stat. Soc.*, pp. 267–288, 1996.
- [29] S.P. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.
- [30] M.G. Christensen and S.H. Jensen, "On compressed sensing and its application to speech and audio signals," in *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*, IEEE, 2009, pp. 356–360.
- [31] D.L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [32] B.D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," *IEEE Trans. on Sig. Proc.*, vol. 47, no. 1, pp. 187–200, 1999.
- [33] I.F. Gorodnitsky and B.D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A Re-weighted minimum norm algorithm," *IEEE Trans. on Sig. Proc.*, vol. 45, no. 3, pp. 600–616, 1997.
- [34] I. Daubechies, R. DeVore, M. Fornasier, and S. Gunturk, "Iteratively re-weighted least squares minimization for sparse recovery," *Commun. Pure Appl. Math.*, vol. 63, no. 1, pp. 1–38, 2009.
- [35] Z. He, A. Cichocki, R. Zdunek, and S. Xie, "Improved FOCUSS method with conjugate gradient iterations," *IEEE Trans. on Sig. Proc.*, vol. 57, no. 1, pp. 399–404, 2009.
- [36] M.E. Davies and L. Daudet, "Sparse audio representations using the MCLT," *Signal processing*, vol. 86, no. 3, pp. 457–470, 2006.
- [37] S. Sardy, A.G. Bruce, and P. Tseng, "Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries," *Journal of computational and graphical statistics*, vol. 9, pp. 361–379, 2000.
- [38] D.L. Donoho and J.M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.
- [39] M. Kowalski, "Sparse regression using mixed norms," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 303–324, 2009.
- [40] T. Blumensath, M. Yaghoobi, and M.E. Davies, "Iterative hard thresholding and l0 regularisation," in *Proc. Int. Conf. Acoust. Speech Sig. Proc.*, 2007.
- [41] M.A.T. Figueiredo and R.D. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Trans. on Image Proc.*, vol. 12, no. 8, pp. 906–916, 2003.

- [42] I. Daubechies, M. Deffrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.
- [43] M. Elad, "Why simple shrinkage is still relevant for redundant representations?," *IEEE Trans. on Inf. Theory*, vol. 52, no. 12, pp. 5559–5569, 2006.
- [44] K.K. Herrity, A.C. Gilbert, and J.A. Tropp, "Sparse approximation via iterative thresholding," in *Proc. Int. Conf. Acoust. Speech Signal Proc.*, 2006, vol. 3, pp. 624–627.
- [45] J.M. Bioucas-Dias and M.A.T. Figueiredo, "A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration," *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2992–3004, 2007.
- [46] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.
- [47] M.S. Lewicki and T.J. Sejnowski, "Learning overcomplete representations," *Neural computation*, vol. 12, no. 2, pp. 337–365, 2000.
- [48] R. Gribonval, "Should penalized least squares regression be interpreted as Maximum A Posteriori estimation?," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2405–2410, May 2011.
- [49] M.D. Plumbley, S.A. Abdallah, T. Blumensath, and M.E. Davies, "Sparse representations of polyphonic music," *Signal Processing*, vol. 86, no. 3, pp. 417–431, 2006.
- [50] K. Kreutz-Delgado, J.F. Murray, B.D. Rao, K. Engan, T.W. Lee, and T.J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Computation*, vol. 15, no. 2, pp. 349–396, 2003.
- [51] D.P. Wipf and B.D. Rao, "Sparse Bayesian learning for basis selection," *IEEE Trans. on Sig. Proc.*, vol. 52, no. 8, pp. 2153–2164, 2004.
- [52] M.E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.
- [53] M.A.T. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1150–1159, 2003.
- [54] G.H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed ℓ_0 norm," *IEEE Trans. on Sig. Proc.*, vol. 57, pp. 289–301, 2009.
- [55] P.J. Wolfe, S.J. Godsill, and W.J. Ng, "Bayesian variable selection and regularization for time-frequency surface estimation," *J. Royal Stat. Soc.*, vol. 66, no. 3, pp. 575–589, 2004.
- [56] C. Fevotte and S.J. Godsill, "Sparse linear regression in unions of bases via Bayesian variable selection," *IEEE Sig. Proc. Lett.*, vol. 13, pp. 441–444, 2006.
- [57] E.I. George and R.E. McCulloch, "Approaches for Bayesian variable selection," *Statistica Sinica*, vol. 7, pp. 339–374, 1997.
- [58] C. Fevotte, B. Torresani, L. Daudet, and S.J. Godsill, "Sparse linear regression with structured priors and application to denoising of musical audio," *IEEE Trans. Audio, Speech, Lang. Proc.*, vol. 16, no. 1, pp. 174–185, 2008.
- [59] R. Gribonval, "Sparse decomposition of stereo signals with matching pursuit and application to blind source separation of more than two sources from a stereo mixture," in *Proc. Int. Conf. Acoust. Speech Signal Proc.*, 2002, vol. 3, pp. 3057–3060.
- [60] L. Daudet, "Sparse and structured decompositions of signals with the molecular matching pursuit," *IEEE Trans. Audio, Speech, Lang. Proc.*, vol. 14, no. 5, pp. 1808–1816, 2006.
- [61] X. Serra and J.O. Smith, "Spectral modeling synthesis: a sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.
- [62] P. Depalle, G. Garcia, and X. Rodet, "Tracking of partials for additive sound synthesis using hidden Markov models," in *Proc. Int. Conf. Acoust. Speech Sig. Proc.*, 1993, vol. 1, pp. 242–245.
- [63] P. Leveau, E. Vincent, G. Richard, and L. Daudet, "Instrument-specific harmonic atoms for mid-level music representation," *IEEE Transactions on Audio Speech and Language Processing*, vol. 16, no. 1, pp. 116–128, 2008.
- [64] B.L. Sturm, J.J. Shynk, and S. Gauglitz, "Agglomerative clustering in sparse atomic decompositions of audio signals," in *Proc. Int. Conf. Acoust. Speech Signal Proc.*, 2008, pp. 97–100.
- [65] M. Kowalski and B. Torr sani, "Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients," *Sig., Image and Video Proc.*, vol. 3, no. 3, pp. 251–264, 2009.
- [66] D. Donoho, "Sparselab website," 2011, <http://sparselab.stanford.edu/>.
- [67] T. Blumensath, "Sparsify website," 2011, <http://users.fmrib.ox.ac.uk/~tblumens/sparsify/sparsify.html>.
- [68] S. Krstulovic and R. Gribonval, "MPTK: Matching Pursuit made tractable," in *Proc. Int. Conf. Acoust. Speech Signal Proc.*, 2006, vol. 3, pp. 496–499.
- [69] C. Kereliuk, "Glablab website and supplementary materials," 2011, <http://www.music.mcgill.ca/~corey/dafx2011>.
- [70] B.L. Sturm, C. Roads, A. McLeran, and J.J. Shynk, "Analysis, visualization, and transformation of audio signals using dictionary-based methods," *Journal of New Music Research*, vol. 38, no. 4, pp. 325–341, 2009.
- [71] S.G. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, Burlington, MA, 3rd edition, 2009.
- [72] A. Adler, V. Emiya, G. Jafari, M. Elad, R. Gribonval, and M.D. Plumbley, "Audio Inpainting," Research report, Mar. 2011.

CONSTRUCTING AN INVERTIBLE CONSTANT-Q TRANSFORM WITH NONSTATIONARY GABOR FRAMES

Gino Angelo Velasco^{*†}, Nicki Holighaus^{*}, Monika Dörfler^{*}, Thomas Grill[‡]

^{*}NuHAG, Faculty of Mathematics, University of Vienna, Austria

[†]Institute of Mathematics, University of the Philippines, Diliman, Quezon City, Philippines

[‡]Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria

{gino.velasco,nicki.holighaus,monika.doerfler}@univie.ac.at,
thomas.grill@ofai.at

ABSTRACT

An efficient and perfectly invertible signal transform featuring a constant-Q frequency resolution is presented. The proposed approach is based on the idea of the recently introduced nonstationary Gabor frames. Exploiting the properties of the operator corresponding to a family of analysis atoms, this approach overcomes the problems of the classical implementations of constant-Q transforms, in particular, computational intensity and lack of invertibility. Perfect reconstruction is guaranteed by using an easy to calculate dual system in the synthesis step and computation time is kept low by applying FFT-based processing. The proposed method is applied to real-life signals and evaluated in comparison to a related approach, recently introduced specifically for audio signals.

1. INTRODUCTION

Many traditional signal transforms impose a regular spacing of frequency bins. In particular, Fourier transform based methods such as the *short-time Fourier transform* (STFT) lead to a frequency resolution that does not depend on frequency, but is constant over the whole frequency range. In contrast, the constant-Q transform (CQT), originally introduced by J. Brown [1, 2], features a frequency resolution dependent on the center frequencies of the windows used for each bin and the center frequencies of the frequency bins are not linearly, but geometrically spaced. In this sense, the principal idea of CQT is reminiscent of wavelet transforms, compare [3]: the Q-factor, i.e. the ratio of the center frequency to bandwidth is constant over all bins and thus the frequency resolution is better for low frequencies whereas time resolution improves with increasing frequency. However, the transform proposed in the original paper [1] is not invertible and does not rely on any concept of (orthonormal) bases. In fact, the number of bins used per octave is much higher than most traditional wavelet techniques would allow for. Furthermore, the computational efficiency of the original transform and its improved versions, [4], may be insufficient.

CQTs rely on perception-based considerations, which is one of the reasons for their importance in the processing of speech and music signals. In these fields, the lack of invertibility of existing CQTs has become an important issue: for important applications such as masking of certain signal components or transposition of

an entire signal or, again, some isolated signal components, the unbiased reconstruction from analysis coefficients is crucial. An interesting and promising approach to music processing with CQT was recently suggested in [5], also cf. references therein.

In the present contribution, we take a different point of view and consider both the implementation and inversion of a constant-Q transform in the context of the *nonstationary Gabor transform* (NSGT). Classical Gabor transform [6, 7] may be understood as a sampled STFT or sliding window transform. The generalization to NSGT was introduced in [8, 9] and allows for windows with flexible, adaptive bandwidths. Figure 1 shows examples of spectrograms of the same signal obtained from the classical sampled STFT (Gabor transform) and the proposed *constant-Q nonstationary Gabor transform* (CQ-NSGT).

If the analysis windows are chosen appropriately, both analysis and reconstruction is realized efficiently with FFT-based methods. The original motivation for the introduction of NSGT was the desire to adapt both window size and sampling density in time, in order to resolve transient signal components more accurately. Here, we apply the same idea in frequency: we use windows with adaptive, compact bandwidth and choose the time-shift parameters dependent on the bandwidth of each window. The construction of the atoms, i.e. the shifted versions of the basic window functions used in the transform, is done directly in the frequency domain, see Sections 2.2 and 3.1. This approach allows for efficient implementation using the FFT, as explained in Section 2.3. To exploit the efficiency of FFT, the signal of interest must be transformed into the frequency domain. For long real-life signals (e.g. signals longer than 10 seconds at a sampling rate of 44100Hz), processing is therefore done on consecutive time-slices, which is a natural processing step in real-time signal analysis¹. The resolution of the proposed CQ-NSGT is identical to that of the CQT and perfect reconstruction is assured by relying on concepts from frame theory, which will be discussed next.

2. NONSTATIONARY GABOR FRAMES

Frames were first mentioned in [10], also see [11, 12]. Frames are a generalization of (orthonormal) bases and allow for redundancy and thus for much more flexibility in design of the signal representation. Thus, frames may be tailored to a specific application

This work was supported by the Vienna Science, Research and Technology Fund (WWTF) project Audio-Miner (MA09-024) and Austrian Science Fund (FWF) projects LOCATIF(T384-N13) and SISE(S10602-N13).

¹If the time-slicing is done using smooth windows with a judiciously chosen amount of zero-padding, no undesired artifacts after modification of the analysis coefficients have to be expected. Mathematical details and error estimates will be given elsewhere.

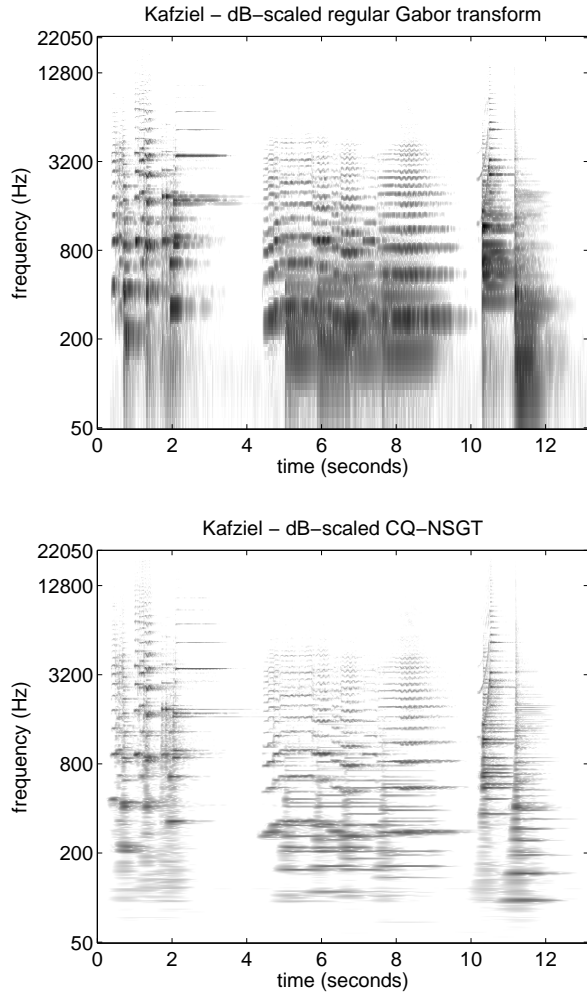


Figure 1: Representations of a musical piece for violin and piano using the classical sampled STFT (Gabor transform) and the CQ-NSGT, respectively. A Hann window of length 1024 samples with a hop-size of 512 samples was used for the Gabor transform, while a minimum frequency of $\xi_{\min} = 50$ Hz at 48 bins per octave was used for the CQ-NSGT.

or certain requirements such as a constant-Q frequency resolution. Loosely speaking, we wish to expand, or represent, a given signal of interest as a linear combination of some building blocks or atoms $\varphi_{n,k}$, with $(n, k) \in \mathbb{Z} \times \mathbb{Z}$, which are the members of our frame:

$$f = \sum_{n,k} c_{n,k} \varphi_{n,k} \quad (1)$$

for some coefficients $c_{n,k}$. The double indexes (n, k) allude to the fact that each atom has a certain location and concentration in time and frequency, compare Figure 2. Frame theory now allows us to determine, under which conditions an expansion (1) is possible and how coefficients leading to stable, perfect reconstruction may be determined.

We introduce the concept of frames for a Hilbert space \mathcal{H} . In a continuous setting, one may think of $\mathcal{H} = L^2(\mathbb{R})$, whereas we

will choose $\mathcal{H} = \mathbb{C}^L$, L being the signal length, for describing the implementation.

2.1. Frames

Consider a collection of atoms $\varphi_{n,k} \in \mathcal{H}$ with $(n, k) \in \mathbb{Z} \times \mathbb{Z}$. Here, n may be thought of as a time index and k as an index related to frequency. We then define the frame operator \mathbf{S} by

$$\mathbf{S}f = \sum_{n,k} \langle f, \varphi_{n,k} \rangle \varphi_{n,k},$$

for all $f \in \mathcal{H}$. Note that, if the set of functions $\{\varphi_{n,k}, (n, k) \in \mathbb{Z} \times \mathbb{Z}\}$ is an orthonormal basis, then \mathbf{S} is the identity operator. If \mathbf{S} is invertible on \mathcal{H} , then the collection $\{\varphi_{n,k}\}, (n, k) \in \mathbb{Z} \times \mathbb{Z}$ is a frame. In this case, we may define a *dual frame* by

$$\gamma_{n,k} = \mathbf{S}^{-1} \varphi_{n,k}.$$

Then, reconstruction from the coefficients $c_{n,k} = \langle f, \varphi_{n,k} \rangle$ is possible:

$$f = \mathbf{S}^{-1} \mathbf{S}f = \sum_{n,k} \langle f, \varphi_{n,k} \rangle \mathbf{S}^{-1} \varphi_{n,k} = \sum_{n,k} c_{n,k} \gamma_{n,k}.$$

2.2. The Case of Painless Nonstationarity

In a general setting, the inversion of the operator \mathbf{S} poses a problem in numerical realization of frame analysis. However, it was shown in [13], that under certain conditions, usually fulfilled in practical applications, \mathbf{S} is diagonal. This situation of *painless non-orthogonal expansions* can now be generalized to allowing for adaptive resolution. Adaptive time-resolution was described in [8, 9], and here we turn to *adaptivity in frequency* in the same manner.

In the sequel, let \mathbf{T}_x denote a time-shift by x , \mathbf{M}_ω denote a frequency shift (or modulation) by ω and $\mathcal{F}f = \hat{f}$ the Fourier transform of f . Let $\varphi_k, k \in \mathbb{Z}$, be band-limited windows, well-localized in time, whose Fourier transforms $\psi_k = \widehat{\varphi_k}$ are centered around possibly irregularly (or, e.g. geometrically) spaced frequency points ξ_k .

Then, we choose frequency dependent time-shift parameters (hop-sizes) a_k as follows: if the support of $\widehat{\varphi_k}$ is contained in an interval of length $|\mathcal{I}_k|$, then we choose a_k such that

$$a_k \leq \frac{1}{|\mathcal{I}_k|} \text{ for all } k.$$

In other words, the time-sampling points have to be chosen dense enough to guarantee this condition. Finally, we obtain the frame members by setting

$$\varphi_{n,k} = \mathbf{T}_{na_k} \varphi_k.$$

Under these conditions on the windows φ_k and the hop-sizes a_k , the frame operator is diagonal in the Fourier domain: since, by unitarity of the Fourier transform [14] and the Walnut representation of the frame operator [15], we have

$$\begin{aligned} \langle \mathbf{S}f, f \rangle &= \sum_{n,k} |\langle f, \mathbf{T}_{na_k} \varphi_k \rangle|^2 = \sum_{n,k} |\langle \hat{f}, \mathbf{M}_{-na_k} \widehat{\varphi_k} \rangle|^2 \\ &= \left\langle \sum_k \frac{1}{a_k} |\widehat{\varphi_k}|^2 \hat{f}, \hat{f} \right\rangle, \end{aligned}$$

the frame operator assumes the following form:

$$\mathbf{S}f = \mathcal{F}^{-1} \left(\sum_k \frac{1}{a_k} |\widehat{\varphi}_k|^2 \widehat{f} \right). \quad (2)$$

See [16, 13, 17] for detailed proofs of the diagonality of the frame operator in the described setting. From (2), it follows immediately that the frame operator is invertible whenever there exist real numbers A and B such that the inequalities

$$0 < A \leq \sum_k \frac{1}{a_k} |\widehat{\varphi}_k|^2 \leq B < \infty \quad (3)$$

hold almost everywhere. In this case, the dual frame is given by the elements

$$\gamma_{n,k} = \mathbf{T}_{na_k} \left[\mathcal{F}^{-1} \left(\widehat{\varphi}_k / \sum_l \frac{1}{a_l} |\widehat{\varphi}_l|^2 \right) \right].$$

2.3. Realization in the Frequency domain

Based on the implementation of nonstationary Gabor frames performing adaptivity in the time domain [9], the above framework permits a fast realization by considering the Fourier transform of the input signal. The transform coefficients $c_{n,k} = \langle f, \varphi_{n,k} \rangle$ take the form

$$c_{n,k} = \langle f, \mathbf{T}_{na_k} \varphi_k \rangle = \langle \widehat{f}, \mathbf{M}_{-na_k} \widehat{\varphi}_k \rangle,$$

and can be calculated, for each k , with an inverse FFT (IFFT) of length determined by the support of $\psi_k = \widehat{\varphi}_k$. Similarly, reconstruction is realized by applying the dual windows $\widehat{\gamma}_k = \widehat{\varphi}_k / \sum_l \frac{1}{a_l} |\widehat{\varphi}_l|^2$ in a simple overlap-add process:

$$\widehat{f} = \sum_{n,k} \langle \widehat{f}, \mathbf{M}_{-na_k} \widehat{\varphi}_k \rangle \mathbf{M}_{-na_k} \widehat{\gamma}_k. \quad (4)$$

3. THE CQ-NSGT PARAMETERS: WINDOWS AND LATTICES

We will now describe in detail the parameters involved in the design of a nonstationary Gabor transform with constant-Q frequency resolution.

The CQT in [1] depends on the following parameters: the window functions, the number of frequency bins per octave, the minimum and maximum frequencies. These parameters determine the Q-factor, which is, as mentioned before, the ratio of the center frequency to the bandwidth. Here, the Q-factor is desired to be constant for all the relevant bins.

Let B and ξ_{\min} denote the number of frequency bins per octave and the desired minimum frequency, respectively. For the proposed CQ-NSGT, we consider band-limited window functions $\varphi_k \in \mathbb{C}^L$, $k = 1, \dots, K$, with center frequencies ξ_k (in Hz) satisfying $\xi_k = \xi_{\min} 2^{\frac{k-1}{B}}$, as in the classical CQT. The maximum frequency ξ_{\max} is restricted to be less than the Nyquist frequency $\xi_s/2$, where ξ_s denotes the sampling frequency. Further, we require the existence of an index K such that $\xi_{\max} \leq \xi_K < \xi_s/2$. We may set $K = \lceil B \log_2(\xi_{\max}/\xi_{\min}) + 1 \rceil$, with $\lceil z \rceil$ denoting the smallest integer greater than or equal to z .

Note that in the CQT, since the frequency spacing in the CQT is geometric, no 0-frequency is present and some high frequency content might not be represented. In the CQ-NSGT, however, there

is freedom to use additional center frequencies, at negligible computational cost, to guarantee perfect reconstruction.

In our current implementation, tailored to (real) audio signals, we consider some symmetry in the frequency domain, and take the following values for the frequency-centers ξ_k :

$$\xi_k = \begin{cases} 0, & k = 0 \\ \xi_{\min} 2^{\frac{k-1}{B}}, & k = 1, \dots, K \\ \xi_s/2, & k = K+1 \\ \xi_s - \xi_{2K+2-k}, & k = K+2, \dots, 2K+1. \end{cases}$$

The bandwidth Ω_k (the support of the window in frequency) of φ_k is set to be $\Omega_k = \xi_{k+1} - \xi_{k-1}$, for $k = 2, \dots, K-1$, which leads to a constant Q-factor $Q = (2^{\frac{1}{B}} - 2^{-\frac{1}{B}})$. To obtain the same Q-factor on the relevant frequency bins, Ω_1 and Ω_K are therefore set to be ξ_1/Q and ξ_K/Q , respectively. Finally, we let $\Omega_0 = 2\xi_1 = 2\xi_{\min}$ and $\Omega_{K+1} = \xi_s - 2\xi_K$. In summary, we have the following values for Ω_k :

$$\Omega_k = \begin{cases} 2\xi_{\min}, & k = 0 \\ \xi_k/Q, & k = 1, \dots, K \\ \xi_s - 2\xi_K, & k = K+1 \\ \xi_{2K+2-k}/Q, & k = K+2, \dots, 2K+1. \end{cases}$$

3.1. Window Choice: Satisfying the Frame Conditions

We now give the details on the windows φ_k to be used such that (3) and hence the frame property is fulfilled.

We use a Hann window \hat{h} that is zero outside $[-1/2, 1/2]$, i.e. a standard Hann window centered at 0 with support of length 1. We obtain the atoms φ_k by translation and dilation of \hat{h} : $\widehat{\varphi}_k[j] = \hat{h}((j\xi_s/L - \xi_k)/\Omega_k)$, $k = 1, \dots, K$, $K+2, \dots, 2K+1$, $j = 0, \dots, L-1$.

For the windows corresponding to the 0 and Nyquist frequencies, we use a plateau-like function \hat{g} , e.g. a Tukey window. We obtain φ_0 and φ_{K+1} by setting $\widehat{\varphi}_k[j] = \hat{g}((j\xi_s/L - \xi_k)/\Omega_k)$, $k = 0, K+1$.

Now, for the collection of time-shifts of the constructed windows a_k , we require $a_k \leq \xi_s/\Omega_k$ in order to satisfy (3). The $\varphi_{n,k}$ are then given by their Fourier transforms as:

$$\widehat{\varphi}_{n,k} = \mathbf{M}_{-na_k} \widehat{\varphi}_k, \quad n = 0, \dots, \lceil \frac{L}{a_k} \rceil - 1.$$

Figure 2 illustrates the time-frequency sampling grid of the set-up with the sampling points taken geometrically over frequency and linearly over time. Given these parameters, the coefficients of the CQ-NSGT are of the form $c_{n,k} = \langle f, \varphi_{n,k} \rangle = \langle \widehat{f}, \widehat{\varphi}_{n,k} \rangle$, $f \in \mathbb{C}^L$. We note that the time-shift parameters can also be fixed to have the same value $a = \min_k \{a_k\}$ and the coefficients obtained from the CQ-NSGT can be put in a matrix of size $\lceil \frac{L}{a} \rceil \times 2(K+1)$.

From the given support condition, the system $\{\widehat{\varphi}_k\}_k$ has an overlap factor of around 1/2. This implies that for the case where $a_k = \xi_s/\Omega_k$, the redundancy of the system is approximately 2.

By construction, the sum $\sum_{m=0}^{2K+1} \frac{L}{a_k} |\widehat{\varphi}_k|^2$ is finite and bounded away from 0. From Sections 2.2 and 2.3, the frame operator is invertible and perfect reconstruction of the signal is obtained from the coefficients $c_{n,k}$ by applying (4).

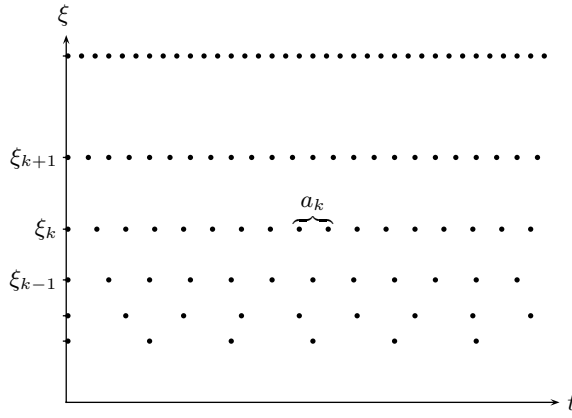


Figure 2: Exemplary sampling grid of the time-frequency plane for a nonstationary Gabor system with resolution evolving over frequency.

signal length L	CQT mean time \pm variance (in seconds)	CQ-NSGT mean time \pm variance (in seconds)
262144	2.41 ± 0.03	0.64 ± 0.00
280789	2.42 ± 0.06	0.68 ± 0.06
579889	3.09 ± 0.06	1.28 ± 0.06
600569 ²	3.13 ± 0.04	1.75 ± 0.04
805686	3.57 ± 0.09	1.51 ± 0.08

Table 1: Comparison of computation time between CQTs and CQ-NSGTs for signals of various lengths over 50 iterations. Parameters for all transforms were $B = 48$ and $\xi_{\min} = 50$ Hz.

4. SIMULATIONS

We now present some experiments comparing the original CQT with the CQ-NSGT in terms of reconstruction error, computation time and (visual) representation of sound signals.

Technical framework: All simulations were done in MATLAB R2009b on a 2 Gigahertz Intel Core 2 Duo machine with 2 Gigabytes of RAM running Kubuntu 9.04. The CQTs were computed using the code published with [5], available for free download at <http://www.elec.qmul.ac.uk/people/anssik/cqt/>. The CQ-NSGT algorithms are available at <http://univie.ac.at/nonstatgab/cqt/>.

For all experiments, ξ_{\max} is taken to be $\xi_K = \xi_{\min} 2^{\frac{K-1}{B}}$, where K is the largest integer such that $2\xi_K < \xi_s$.

4.1. Reconstruction Errors

The theoretical results, stating that the CQ-NSGT allows for perfect reconstruction, are confirmed by our experiments. For five test signals and various transform parameters, the relative reconstruction error

²Since the proposed method relies on an initial FFT of length L , a prime valued signal length may give a longer computation time.

Bins per octave B	CQT mean time \pm variance (in seconds)	CQ-NSGT mean time \pm variance (in seconds)
12	0.95 ± 0.01	0.36 ± 0.00
24	1.44 ± 0.02	0.44 ± 0.00
48	2.42 ± 0.03	0.65 ± 0.00
96	4.50 ± 0.23	1.09 ± 0.15

Table 2: Comparison of computation time between CQTs and CQ-NSGTs of the Glockenspiel signals, varying the number of bins per octave. Values were obtained over 50 iterations. The minimum frequency ξ_{\min} was chosen at 50 Hz.

tion error

$$e_{\text{rec}} = \sqrt{\frac{\sum_{j=0}^{L-1} |f[j] - f_{\text{rec}}[j]|^2}{\sum_{j=0}^{L-1} |f[j]|^2}}$$

was calculated. With ξ_{\min} between 10 Hz and 130 Hz and B from 12 to 192, the largest reconstruction error of the CQ-NSGT algorithm was slightly smaller than $1.6 \cdot 10^{-15}$, perfect reconstruction up to numerical precision. For comparison, it was shown in [5] that a CQT with reasonable amounts of redundancy and bins per octave can be inverted with a relative error of 10^{-3} . This might not be enough for high-quality applications.

4.2. Computation Time and Computational Complexity

The required time for construction of the transform atoms and computation of the corresponding coefficients was measured for audio signals of roughly 6 to 18 seconds length, at a sampling rate of 44.1 kHz. Each experiment was repeated 50 times, the results are listed in Table 1. We note that for all signals, the CQ-NSGT is faster than the CQT implementation proposed in [5] by a considerable factor.

Our approach is still of complexity $\mathcal{O}(L \log L)$, though, and the advantage over the CQT decreases for longer signals. Each frequency channel's time samples are acquired by means of sampled IFFT from the Fourier transform of the input signal, multiplied with the corresponding window. Therefore, a preliminary full length FFT is necessary.

More explicitly, we assume $\widehat{\varphi}_k$ to have support of length M_k and we denote by N_k the corresponding IFFT length. Let $N = \max_k \{N_k\}$, i.e. the maximum IFFT-length, and we have $M_k \leq N_k \leq N$, since we only consider the painless case. Consequently, the number of operations is as follows:

1. FFT: $\mathcal{O}(L \cdot \log(L))$.
2. Windowing: M_k operations for the k -th window.
3. IFFT: $\mathcal{O}(N_k \cdot \log(N_k))$ for the k -th window.

The number of frequency channels $2K + 2$ is independent of L , since it is determined directly from the transform parameters. Thus, M_k and N_k are L -dependent and the computational complexity of the discrete CQ-NSGT is $\mathcal{O}(L \log L)$.

In applications, the dual windows are constructed directly on the frequency side and the painless case construction involves multiplication of the window functions by the inverse of a diagonal matrix, resulting in $\mathcal{O}(2 \sum_{k=0}^{2K+1} M_k) = \mathcal{O}(L)$ operations. Finally, the inverse CQ-NSGT has numerical complexity

$\mathcal{O}(L \cdot \log(L))$, since it entails computing for the FFT of each coefficient vector, multiplying with the corresponding dual windows and, after evaluating the sum, computing a length L IFFT.

We note that linear computation time may be achieved by processing the signal in a suitable piecewise manner. Some experiments on that matter have been conducted, but the details of this procedure exceed the scope of this paper and are intended to be part of a future contribution.

In a second experiment, CQT and CQ-NSGT coefficients of the shortest sample, a Glockenspiel signal, were computed for several numbers of bins per octave. We note that the complexity of the algorithm for the CQ-NSGT is linear in B . The results, listed in Table 2, illustrate that the advantage of the CQ-NSGT algorithm increases for large numbers of bins.

4.3. Visual Representation of Sound Signals

The spectral representation provided by CQT has several desirable properties, e.g. the logarithmic frequency scale resolves musical intervals in a similar way, independent of absolute frequencies. These properties are still present in the CQ-NSGT, in fact, its visual representation is practically identical to that of classical CQT as illustrated by Figure 3 for the exemplary case of the Glockenspiel signal. Figure 4 shows the CQ-NSGT of two additional music signals, further illustrating that even highly complex signals are nicely resolved by the proposed transform, similar to CQT.

5. EXPERIMENTS ON APPLICATIONS

Our experiments show applications of the CQ-NSGT in musical contexts, where the property of a logarithmic frequency scale renders the method often superior to the traditional STFT. Corresponding sound examples can be found at <http://univie.ac.at/nonstatgab/cqt/>.

5.1. Transposition

A useful property of continuous constant-Q decompositions is the fact that the transposition of a harmonic structure, like a note including overtones, corresponds to a simple translation of the logarithmically scaled spectrum. Approximately, this is also the case for the finite, discrete CQ-NSGT. In this experiment, we transposed a piano chord simply by shifting the inner frequency bins accordingly. By inner frequency bins, we refer to all bins with constant Q-factor. This excludes the 0-frequency and Nyquist frequency bins. The onset portion of the signal has been damped, since inharmonic components, such as transients, produce audible artifacts when handled in this way. In Figure 5, we show spectrograms of the original and modified chords, shifted by 20 bins. This corresponds to an upwards transposition by 5 semitones.

5.2. Masking

In the masking experiment, we show that the perfect reconstruction property of CQ-NSGT can be used to cut out components from a signal by directly modifying the time-frequency coefficients. The advantage of considerably higher spectral resolution at low frequencies (with a chosen application-specific temporal resolution at higher frequencies) compared to the STFT, makes the CQ-NSGT a very powerful, novel tool for masking or isolating time-frequency components of musical signals. Our example shows in Figure 6 a mask for extracting – or inversely, suppressing – a note from

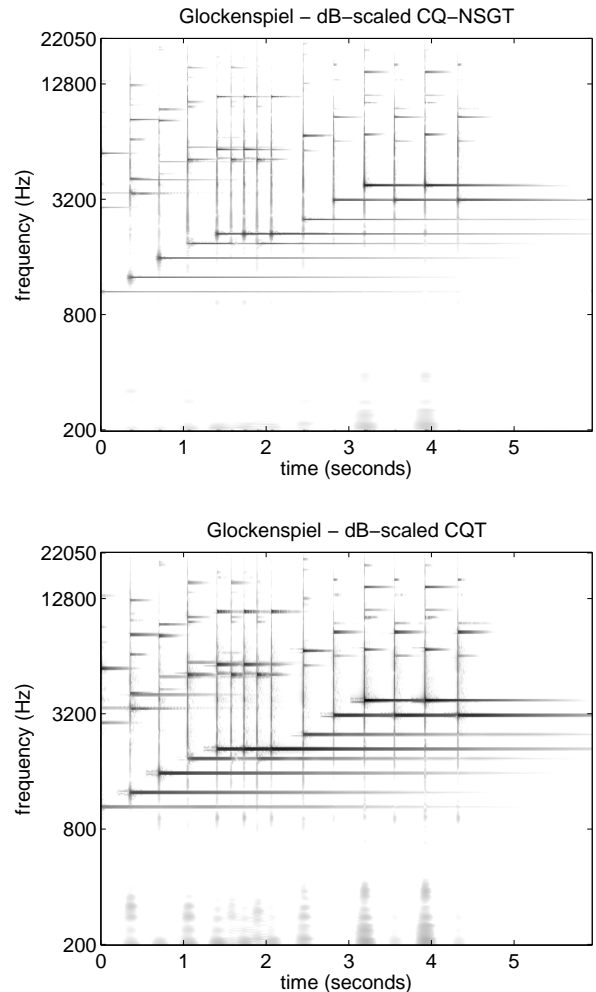


Figure 3: Representations of the Glockenspiel signal using the CQ-NSGT and the original CQT. The transform parameters were $B = 48$ and $\xi_{\min} = 200$ Hz.

the Glockenspiel signal depicted in Figure 3. The mask was created as a gray-scale bitmap using an ordinary image manipulation program and then resampled in order to conform to the irregular time-frequency grid of the CQ-NSGT. Figure 6 shows the mask spectrogram, along with the spectrograms of the synthesized, processed signal and remainder.

6. SUMMARY AND PERSPECTIVES

We presented a constant-Q transform, based on nonstationary Gabor frames, that is computationally efficient and allows for perfect reconstruction. The described framework can easily be adapted to other perceptive frequency scales (e.g. mel or Bark scale) by choosing appropriate dictionaries.

The possibility of overcoming the difficulties that stem from dependence of the proposed transform on the signal length, e.g. by piecewise processing, is currently under investigation. This will further reduce computational effort and enable the use of a single family of frame elements for signals of arbitrary length.

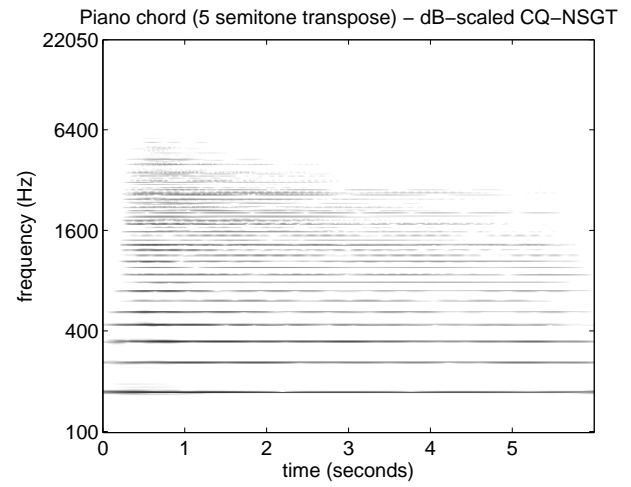
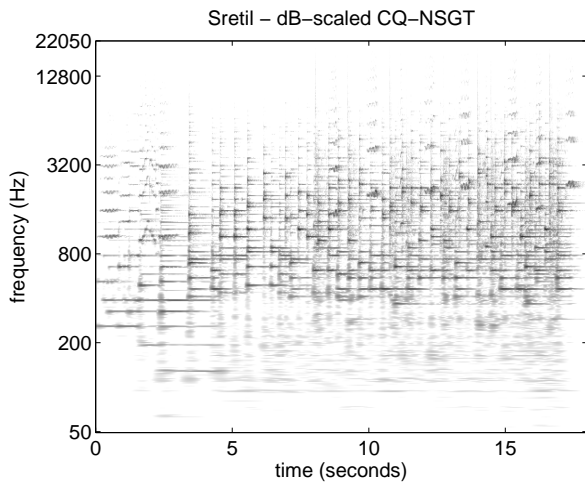
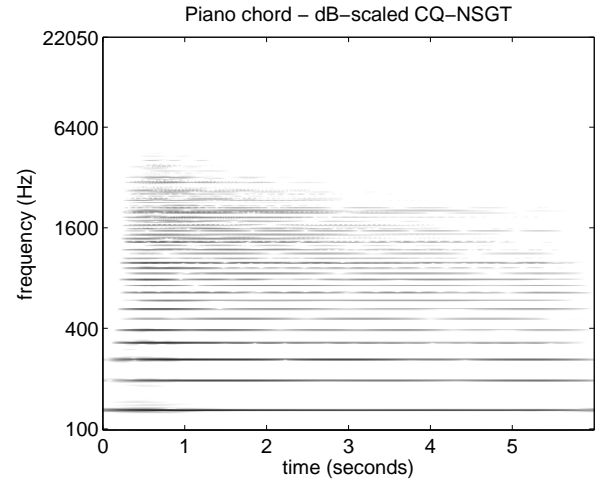
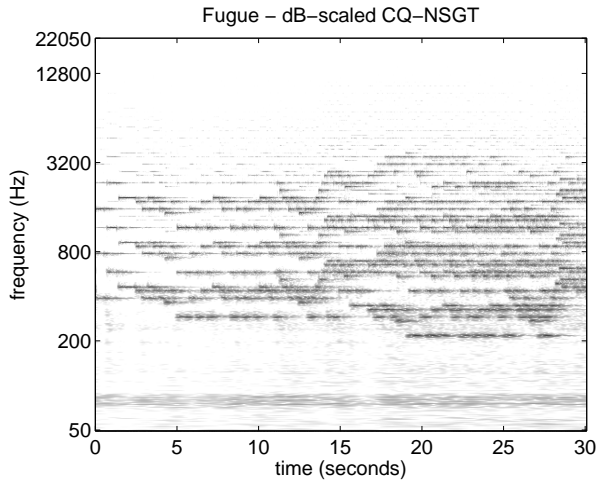


Figure 4: Representations of a pipe organ and piano solo recordings, respectively, using the CQ-NSGT. The transform parameters were $B = 48$ and $\xi_{\min} = 50$ Hz.

Figure 5: Piano chord signal and upwards transposition by 5 semitones, corresponding to a circular shift of the inner bins by 20. The transform parameters were $B = 48$ and $\xi_{\min} = 100$ Hz.

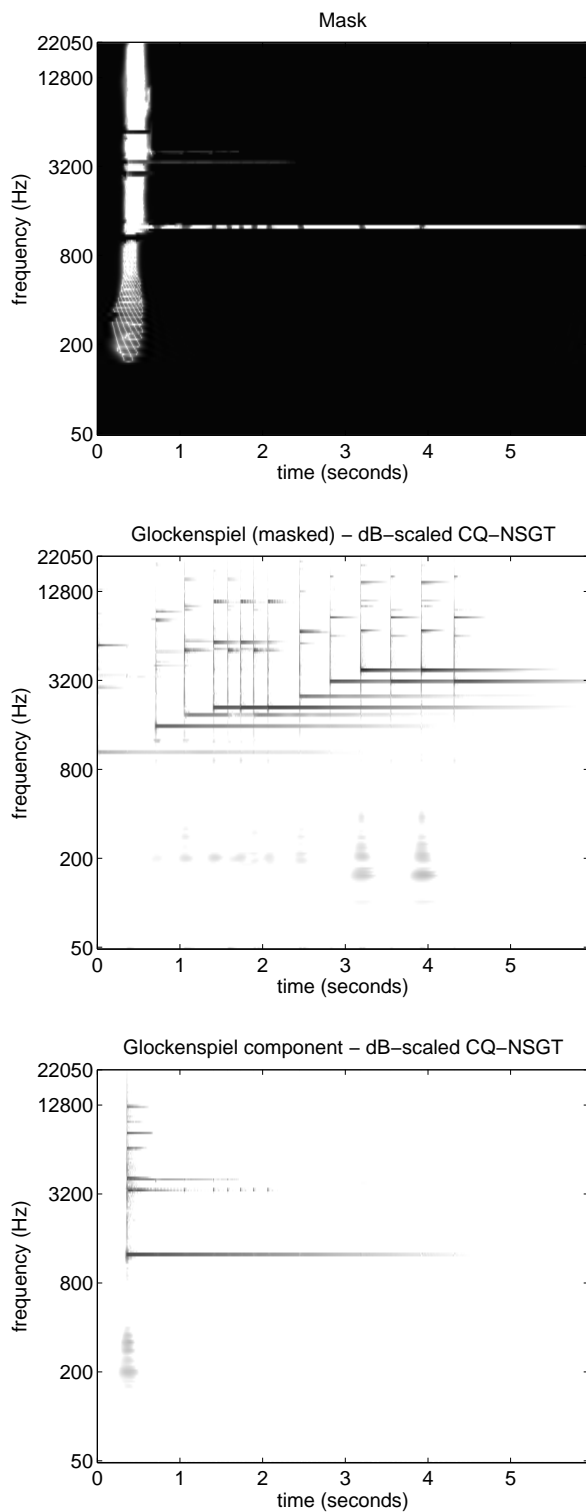


Figure 6: Note extraction from the Glockenspiel signal by masking. The CQ-NSGT coefficients of the Glockenspiel signal were weighted with the mask shown on top. The remaining signal and extracted component are depicted in the middle and bottom respectively. The transform parameters were $B = 24$ and $\xi_{\min} = 50$ Hz.

7. REFERENCES

- [1] J. C. Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. Amer.*, vol. 89, no. 1, pp. 425–434, 1991.
- [2] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, 2009.
- [3] İ. Bayram and I. W. Selesnick, "Frequency-domain design of overcomplete rational-dilation wavelet transforms," *IEEE Trans. Signal Process.*, vol. 57, no. 8, pp. 2957–2972, 2009.
- [4] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *J. Acoust. Soc. Am.*, vol. 92, no. 5, pp. 2698–2701, 1992.
- [5] C. Schörkhuber and A. Klapuri, "Constant-Q toolbox for music processing," in *Proceedings of SMC Conference 2010*, 2010.
- [6] H. G. Feichtinger and T. Strohmer, *Gabor Analysis and Algorithms. Theory and Applications.*, Birkhäuser, 1998.
- [7] H. G. Feichtinger and T. Strohmer, *Advances in Gabor Analysis*, Birkhäuser, 2003.
- [8] F. Jaillet, *Représentation et traitement temps-fréquence des signaux audio-numériques pour des applications de design sonore*, Ph.D. thesis, Université de la Méditerranée Aix-Marseille II, 2005.
- [9] F. Jaillet, P. Balazs, M. Dörfler, and N. Engelputzer, "Non-stationary Gabor frames," in *SAMPTA'09, International Conference on SAMpling Theory and Applications*, 2009, pp. 227–230.
- [10] R. J. Duffin and A. C. Schaeffer, "A class of nonharmonic Fourier series," *Trans. Amer. Math. Soc.*, vol. 72, pp. 341–366, 1952.
- [11] A. Chebira and J. Kovačević, "Life beyond bases: The advent of frames (Part I and II)," *IEEE Signal Processing Magazine*, vol. 24, no. 4 - 5, pp. 86–104, 115–125, 2007.
- [12] O. Christensen, *An Introduction To Frames And Riesz Bases*, Birkhäuser, 2003.
- [13] I. Daubechies, A. Grossmann, and Y. Meyer, "Painless nonorthogonal expansions," *J. Math. Phys.*, vol. 27, no. 5, pp. 1271–1283, May 1986.
- [14] W. Rudin, *Real and Complex Analysis (Third ed.)*, Singapore: McGraw Hill, 1987.
- [15] D. F. Walnut, "Continuity properties of the Gabor frame operator," *J. Math. Anal. Appl.*, vol. 165, no. 2, pp. 479–504, 1992.
- [16] M. Dörfler, "Time-frequency analysis for music signals. A mathematical approach," *Journal of New Music Research*, vol. 30, no. 1, pp. 3–12, 2001.
- [17] P. Balazs, M. Dörfler, F. Jaillet, N. Holighaus, and G. A. Velasco, "Theory, implementation and applications of non-stationary Gabor Frames," *preprint*, submitted, 2011.

MULTIRESOLUTION STFT PHASE ESTIMATION WITH FRAME-WISE POSTERIOR WINDOW-LENGTH DECISION

Volker Gnann and Martin Spiertz

Institut für Nachrichtentechnik, RWTH Aachen University
D-52056 Aachen, Germany
{gnann, spiertz}@ient.rwth-aachen.de

ABSTRACT

This paper presents an extension to the dual-window-length Real-Time Iterative Spectrogram Inversion phase estimation algorithm (RTISI). Instead of a transient detection in advance, the phase estimator itself determines the correct window length when the phase information for all window lengths have already been estimated. This way, we get significant improvements compared with the previous method. Additionally, we extend this estimator to configurations with three or more window lengths.

1. INTRODUCTION

The reconstruction of missing phase information is an important step to get an audio signal from a magnitude short-time Fourier transform (STFT) spectrogram and enables audio effects to work just on the spectrogram magnitude. However, the quality of this approach suffers from the time/frequency resolution tradeoff, similarly to other spectrogram-based audio manipulation methods. In audio coding, window switching is a well-known method to improve this tradeoff. In a previous paper, [1], we already presented an implementation of a phase estimator using window switching between two analysis windows.

Lukin and Todd [2] proposed a different method to perform arbitrary spectrogram-based audio effects with different spectrogram lengths: They process audio frame-wise in parallel with different window length and decide afterwards, frame by frame, which process has performed best. However, this approach needs the knowledge of the phase information. This paper proposes a method to estimate it.

Dual-resolution phase estimation with window switching has some problems which do not occur with the Lukin/ Todd approach:

- Errors in transient detection lead to a sub-optimal time/frequency resolution for the recent audio frame. The decision which resolution is correct is not trivial.
- Algorithms which modify the resulting spectrum do not get the whole spectrogram for processing. Instead,

they get only the frames the transient detector has allocated to them. This can be a disadvantage, e. g. when they need complete statistics for correct processing.

- It is not guaranteed that an optimal decision before the spectrum modification remains optimal during the modification.

This motivates us to create a multi-window-length STFT phase estimator not based on window switching, but on parallel processing, similar to the Lukin/Todd processing scheme. One important difference: In the original paper [2], a coefficient mixing is proposed to determine the final signal. Since such a mixing can lead to phase cancellations, we perform a strong decision for every frame which time-frequency resolution is most appropriate.

As phase estimation algorithm, we use the Real-Time Iterative Spectrogram Inversion (RTISI) [3], with the improvements from [4]. To our knowledge, there are no other phase estimators available that could handle multi-window-length spectrograms. RTISI itself is a localized variation of the classic Griffin/Lim algorithm [5]. An alternative would be the phase estimator of Le Roux et al [6]. Unfortunately, unlike RTISI, this phase estimator operates only in frequency domain, while we need the time-domain representation of intermediate signals to synchronize the buffers (see below for details). Unlike in [1], this synchronization must happen after every committed frame to avoid drifting phase estimations.

This paper is organized as follows. Section 2 gives an overview over the whole processing scheme. Section 3 introduces the phase estimation and synchronization scheme. Section 4 shows how the best estimation is selected. Section 5 generalizes the scheme to more window lengths than two. Section 6 contains the experiments and results. Here, the new method is also compared with [1]. The paper finishes with the conclusions.

2. PROCESSING OVERVIEW

Figure 1 shows how the overall processing is performed. From the original waveform $s[n]$, we generate two magnitude spectrograms with different window lengths L_1 and L_2 and

the same overlap (typically 75%) using the STFT. We call the resulting hop lengths S_1 and S_2 . The ratio L_1/L_2 must be a power of two. As window function, we use the scaled Hamming window from [5].

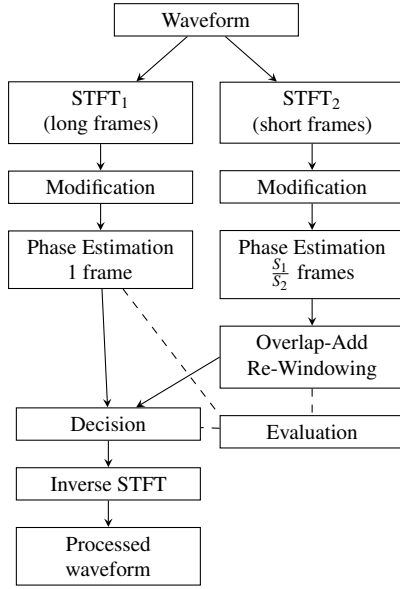


Figure 1: General overview. The evaluation step compares all phase estimations frame-wise with all spectrograms and decides with a minimax principle.

Those magnitude spectrograms can be modified in an arbitrary way. After that, two RTISI algorithms [3] re-estimate the phase spectra. As explained in Section 3, one RTISI buffer is used for each window length, respectively. For each frame of length L_1 except the first one, $\frac{S_1}{S_2}$ frames of length L_2 are estimated, so that the number of estimated samples is the same.

The result of the short-window-length phase estimation is re-windowed such that it is implicitly windowed with the long window. When the estimated samples are available for both window lengths, we rate them using the procedure explained in Section 4 to get the best estimation. The best estimated frame is committed, so that a final overlap-add procedure can collect the committed frames to construct the final modified audio signal.

To ensure that the initialization data of the RTISI buffers are the same, the data of the chosen RTISI buffer must be copied to the other one every time when a frame is committed. Finally, the buffers are set to the next frame. Thus, the next long-window frame, and the next $\frac{S_1}{S_2}$ short-window-frames are processed, and so on.

3. PHASE ESTIMATION

Central data structure for our phase estimator is a combination of two two-dimensional buffers $^{\text{long}}\mathbf{B}$ and $^{\text{short}}\mathbf{B}$, one for each window length. The buffers are basically illustrated in Figure 3 with two modifications: The number of rows in the long-window-length RTISI buffer is actually 7, with the commit frame in the center. Additionally, each buffer row stores the target magnitude spectrum.

Mathematically, we can interpret these buffers as two-dimensional arrays (not matrices) (\mathbf{B}_{MN}) , which are illustrated in Figure 2. On each buffer \mathbf{B} , we define the following operations. To help understanding, operations returning a complete row are overlined, whereas operations returning only a vector of the window size are marked with a degree symbol. The symbol i denotes a row index, a denotes an external vector.

- Addition, Subtraction, Multiplication, Division are defined element-wise.
- $\mathbf{B}.\mathring{\text{CROP}}(i, a)$: Shortens the row vector (a_1, \dots, a_N) to (a_l, \dots, a_r) , with $l := iS$ and $r := l + L$.
- $\mathbf{B}.\overline{\text{GET}}(i)$: Returns the row vector (B_{i1}, \dots, B_{iN}) .
- $\mathbf{B}.\mathring{\text{GET}}(i) := \mathbf{B}.\mathring{\text{CROP}}(i, \mathbf{B}.\overline{\text{GET}}(i))$
- $\mathbf{B}.\mathring{\text{SET}}(i, a)$: Sets the row vector (B_{i1}, \dots, B_{iN}) to $\mathbf{B}.\mathring{\text{CROP}}(i, a)$, with l and r defined as for the $\mathring{\text{CROP}}$ operation.
- $\mathbf{B}.\overline{\text{SET}}(i, a)$: Sets the row vector (B_{i1}, \dots, B_{iN}) to a .
- $\mathbf{B}.\overline{\text{SUM}}$: Calculates the sum of the matrix rows as a row vector $(\sum_{i=1}^M B_{i1}, \dots, \sum_{i=1}^M B_{iN})$.
- $\mathbf{B}.\mathring{\text{SUM}}(i) := \mathbf{B}.\mathring{\text{CROP}}(i, \mathbf{B}.\overline{\text{SUM}}(i))$

For convenience, we also associate following functions with the buffer which return additional data:

- $\mathbf{B}.\text{MAG}(i)$: Returns the target magnitude spectrum of row i .
- $\mathbf{B}.\text{W}$: Returns the discrete window function the magnitude spectra are inherently calculated with.

RTISI, introduced in [3], is an online-capable algorithm, which works on a frame-after-frame basis. The phase estimation consists of two steps: an initialization and several iterations of an (improving) update rule.

The initialization depends on the window length: The short-window buffer is always initialized with zeros. The long-window buffer is initialized with a propagated phase of the previous frames, as proposed in [4], to exploit the phase continuity of steady-state signals.

The update rule is the same on both buffers. It is processed on a buffer row and works as follows:

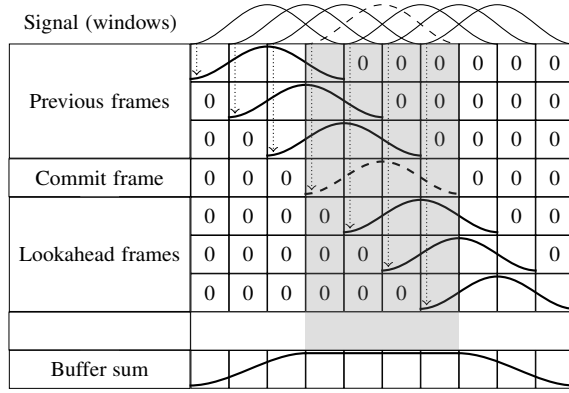


Figure 2: Single phase estimation buffer. Every sketched cell contains S elements, whereas S denotes the hop length between adjacent frames.

- Calculate the sum frequency spectrum for the current row i by taking the according part of the buffer sum, re-windowing this part and applying an Discrete Fourier Transform.
- Project all coefficients of this spectrum onto the unit circle. The result is equivalent to the phase. Multiply this result with the target magnitude (M-Constrained transform).
- Transform the result of this combination into the time domain, window it, and store it back into the current row.

In short:

$$\mathbf{B}.\mathbf{S}\mathbf{\hat{E}}\mathbf{T}(i, \mathbf{B}.\mathbf{W} \cdot \left(\text{IDFT} \left(\mathbf{B}.\mathbf{M}\mathbf{A}\mathbf{G}(i) \cdot \frac{\text{DFT}(\mathbf{B}.\mathbf{S}\mathbf{U}\mathbf{M}(i))}{|\text{DFT}(\mathbf{B}.\mathbf{S}\mathbf{U}\mathbf{M}(i))|} \right) \right)), \quad (1)$$

where DFT denotes the Discrete Fourier Transform, and IDFT its inverse.

The order of the buffer row updates is determined by the energy of the rows — first the loudest row is updated, then the second loudest, and so on, until all rows between the last and the commit row have been processed [4]. After a certain number of iterations, the frame is committed, and the buffer is synchronized to the next audio frame.

After each commit of the long-window buffer and $\frac{S_1}{S_2}$ commits of the short-window buffer, we must decide which configuration is better (see Section 4). After that, the result is copied from the “winning” buffer $^{\text{src}}\mathbf{B}$ to the “losing” buffer $^{\text{tgt}}\mathbf{B}$ as follows:

- Calculate the buffer sum.
- Divide the buffer sum by the sum of the squared window functions for all rows so that the buffer sum is implicitly windowed with a rectangle window.

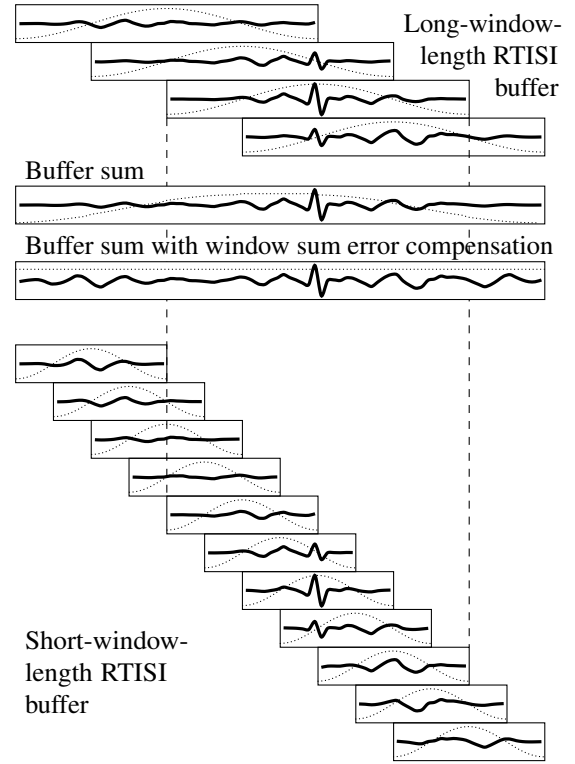


Figure 3: Two RTISI buffers with different window lengths without target magnitude spectra and without look-ahead frames. To transport audio data between the buffers, the algorithm calculates the buffer sum, compensates the window sum error, and windows the result for each target buffer row. The dotted lines denote the window function the audio data in the buffer are implicitly multiplied with.

- For each target buffer row, window the according part of the buffer sum. Copy the windowed buffer sum part into the target row.

To express this mathematically, we consider the RTISI buffers $^{\text{long}}\mathbf{W}$ and $^{\text{short}}\mathbf{W}$ which are filled with the squared windows such that $^{\text{long}}\mathbf{W}.\mathbf{G}\mathbf{\hat{E}}\mathbf{T}(i) = ^{\text{long}}\mathbf{B}.\mathbf{W}^2$ for all i , and $^{\text{short}}\mathbf{W}.\mathbf{G}\mathbf{\hat{E}}\mathbf{T}(i) = ^{\text{short}}\mathbf{B}.\mathbf{W}^2$ for all i . We denote the “window buffers” with the source and target window length as $^{\text{src}}\mathbf{W}$ and $^{\text{tgt}}\mathbf{W}$, respectively. Then the synchronization becomes

$$^{\text{tgt}}\mathbf{B}.\mathbf{S}\mathbf{\hat{E}}\mathbf{T}(i, \left(^{\text{tgt}}\mathbf{W}.\mathbf{G}\mathbf{\hat{E}}\mathbf{T}(i) \cdot ^{\text{tgt}}\mathbf{B}.\mathbf{C}\mathbf{R}\mathbf{O}\mathbf{P}(i, \frac{^{\text{src}}\mathbf{B}.\mathbf{S}\mathbf{U}\mathbf{M}}{^{\text{src}}\mathbf{W}.\mathbf{S}\mathbf{U}\mathbf{M}}) \right)) \quad (2)$$

for all i .

4. DETERMINING THE OPTIMAL WINDOW SIZE

To find the optimal window size, we have to compare the phase estimation for each window length with the magnitude spectrograms, also for each window length. Let \tilde{M} be the position index of the commit frame. As comparison criterion, we use the signal-to-error ratio (in dB) in the magnitude spectrogram domain:

$$\text{SER} = 10 \log_{10} \frac{\sum_{m=\tilde{M}-\xi}^{\tilde{M}+\xi} \left(\sum_{k=0}^{L-1} |X[mS, k]|^2 \right)}{\sum_{m=\tilde{M}-\xi}^{\tilde{M}+\xi} \left(\sum_{k=0}^{L-1} (|X[mS, k]| - |X'[mS, k]|)^2 \right)} \quad (3)$$

From the RTISI buffer with the window length L_u to evaluate, we calculate the actual magnitude spectrogram X' from the buffer sum on the commit frame $\pm \xi$ additional rows; see Section 6.2 why a choice $n = 2$ can be considered optimal. For this spectrogram calculation, we use the window length L_v of the target spectrogram to compare. We denote this SER as $\text{SER}_{u,v}$ which means: "Calculate the signal-to-error ratio of the spectrogram of the phase estimation with the length L_u , but use length L_v for the spectrogram calculation and thus set $S = L_v/4$ for 75% overlap. Compare the resulting spectrogram with the target magnitude stored in the RTISI buffer of the length L_v ."

The four signal-to-error ratios $\text{SER}_{u,v}$ for one long and one short window can be summarized in a matrix:

$$\begin{pmatrix} \text{SER}_{1,1} & \text{SER}_{1,2} \\ \text{SER}_{2,1} & \text{SER}_{2,2} \end{pmatrix} \quad (4)$$

To get a final decision from these four SER values, we derive a minimax approach comparable to the Hausdorff distance [7] as follows: Low SER values for a *short* reference window length L_v correspond to artifacts manifesting in the time domain, usually time-domain smearing. These artifacts are audible especially in transient regions. On the other hand, low SER values for a *long* reference window length correspond to errors in the frequency domain, e. g. modulation effects. They are usually audible at steady-state signals, especially at low frequencies. Since our goal is the reduction of audible artifacts, we dump the estimation with the worst SER and favor the time-domain signal estimation which does *not* produce this worst SER.

As the SER is also the optimization criterion for RTISI, the SERs on the main diagonal of the matrix are optimized, so they are always better than other SER values. For that reason, the worst SER is either $\text{SER}_{1,2}$ or $\text{SER}_{2,1}$. Thus, we can simply decide: If $\text{SER}_{1,2} > \text{SER}_{2,1}$, we choose L_1 , otherwise L_2 .

5. GENERALIZATION TO MORE WINDOW LENGTHS

The extension of this approach to multiple window lengths is now straightforward. For every used window length except the longest one, we employ one separate RTISI buffer and treat it as a short-window case. The longest-window-length buffer is treated as the long-window-length buffer in the previous sections.

To determine the correct window length after estimation, we also generalize the avoid-the-worst-SER approach. For every window length, we compare the phase estimation result with each reference spectrogram and find the worst SER. We choose the window length where the worst SER is best:

$$U = \underset{u}{\operatorname{argmax}} \left(\min_v (\text{SER}_{L_u, L_v}) \right) \quad (5)$$

6. EXPERIMENTS AND RESULTS

In order to evaluate these improvements, we use a test set based on the Sound Quality Assessment Material (SQAM) from the EBU [8]. Our test set consists of 70 files containing speech, singing vocals, and instruments. The sampling frequency is 48 kHz. For spectrogram generation and phase estimation, we use the scaled Hamming window from [5] with $L = 4S$, yielding an overlap of 75%. For phase estimation, we use RTISI with three look-ahead frames (see also the remarks to Fig. 3). As evaluation measure, we use the mean signal-to-error ratio (in dB) of the magnitude spectrograms of the phase-reestimated signal versus the original, respectively, as defined in Equation (3).

Like in Section 4, this SER measure operates on STFT magnitudes and thus depends on its own window length. Also, this SER window length determines the operating point of the *measured* time-frequency resolution tradeoff. To analyze the phase estimation performance over the whole range of time/frequency resolutions, the SER values are plotted against this window length. A high-quality phase reestimation should achieve high SER values for all window lengths to show that it avoids artifacts in both time and frequency domain.

6.1. General Results

Figure 4 shows the SER results of a single-resolution RTISI phase reestimation with the window lengths 512, 1024, and 2048 samples, respectively. Note the peaks at the window lengths; here the evaluation criterion is identical to the optimization criterion. Additionally, this figure shows the results of dual-length phase estimation with transient detection and with post-estimation decision. As a transient detection measure, we use the maximal energy compaction principle described in [2]. We see that our post-estimation decision

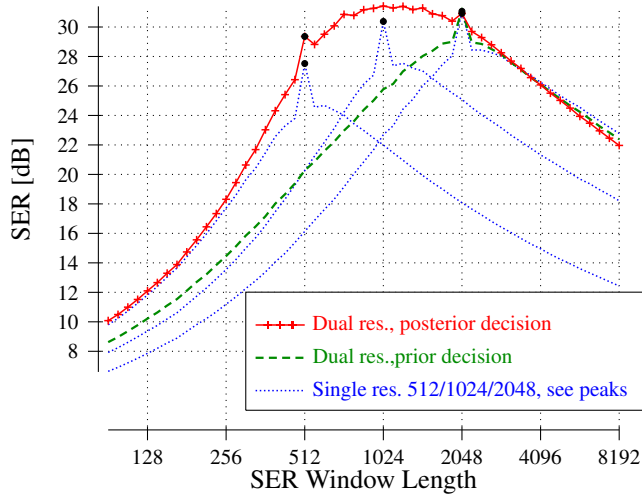


Figure 4: Average SER for different phase estimation methods on the standard EBU set against the measure window length. The crosses on the solid (red) line denote the measure points. The peaks marked with black-filled balls result from the fact that, on this window length, the evaluation criterion matches the optimization criterion.

outperforms the transient detection approach except on very large windows. It delivers also better results than each of the single-resolution estimations, even when evaluation and estimation window length match. The outlier peaks here are smaller than in the single-resolution case; an obvious reason is that the optimization result is a compromise between two window lengths.

6.2. Number of frames to analyze in a buffer

Figure 5 shows which number of frames ξ to analyze in the buffer should be chosen. As a general rule we can follow, the higher the number of frames, the better the estimation. An interesting exception arises when we choose the complete buffer (seven frames, e. g. commit frame ± 3 frames). Here the results are *worse* compared with the five-frames evaluation.

To explain this decline, we recall that all magnitude spectra are implicitly windowed with $w[n]$. In the state of convergence, the M-constrained transform does not change anything, so due to the following windowing step the data in the buffer are implicitly windowed with $w^2[n]$. This window is compensated in the rectification step.

If the state of convergence is not achieved, the M-constrained transform *does* perform changes and tends to neutralize the previous windowing step. As a result, after the following windowing step, the data in the buffer are implicitly windowed with something between $w[n]$ and $w^2[n]$. The rectification over-compensates this windowing step, thus dis-

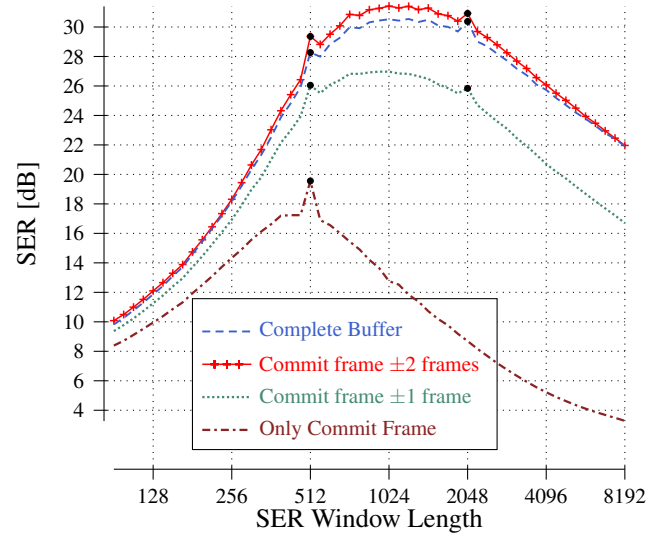


Figure 5: Average SER with respect to the number of long-size buffer frames to take into account for minimax evaluation.

torting the buffer sum especially at the edges. When this edge is also considered for determining the correct window size, this error leads in some cases to wrong decisions and finally to lower SER values than possible.

6.3. Distance between Window Lengths

Figure 6 shows the influence of the difference between the window lengths. One window length – 2048 samples – is fixed, the other one varies between 256 and 2048 samples. As we can expect, the temporal resolution becomes better with increasing difference. On the other hand, in this case the SER values above a window length of 2048 become worse. We can conclude that the time/frequency resolution tradeoff in a certain way also holds for multi-resolution phase estimation.

6.4. Extension to multiple window lengths

The results of using more than two window lengths are presented in Figure 7 and 8. Generally, it makes sense to compare the 512/2048 dual-resolution curve with the 512/1024/2048 triple-resolution curve, and the 256/4096 dual-resolution curve with the 5-resolution curve. In both cases, the SER for the reference window lengths between L_1 and L_2 are better when more than two resolutions are considered. On the other hand, the SER at very long windows becomes smaller. A comparison between the 5-resolution and dual-resolution 256/2048 samples (green, solid curve) shows that the SER gain for the 5-resolution for long window lengths is also limited; for short window lengths the dual-resolution is even better.

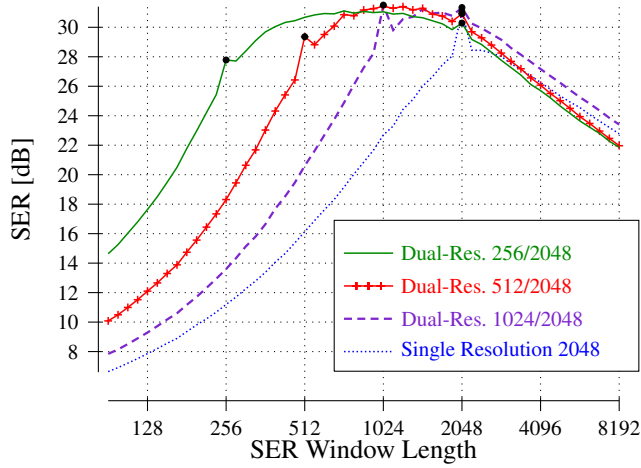


Figure 6: SER for dual window length with one window length (2048 samples) kept fixed.

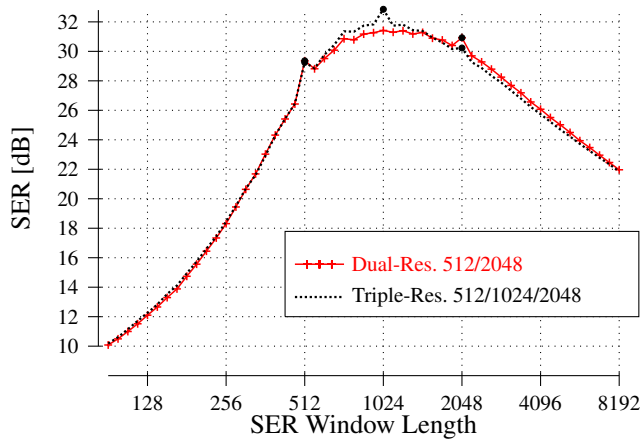


Figure 7: Average SER for triple resolution.

7. CONCLUSIONS

We have generalized the method of [1] to a signal processing scheme that allows switching between different window lengths without relying on a (better or worse) transient detector. Additionally this scheme allows the usage of more than two window lengths. Applications for this technique are all algorithms which operate on magnitude-spectrums only, like time/pitch modification or source separation. While the new approach performs significantly better than the previous method [1], the results for three or more window lengths indicate that in this case the window length decision method gives room for additional research.

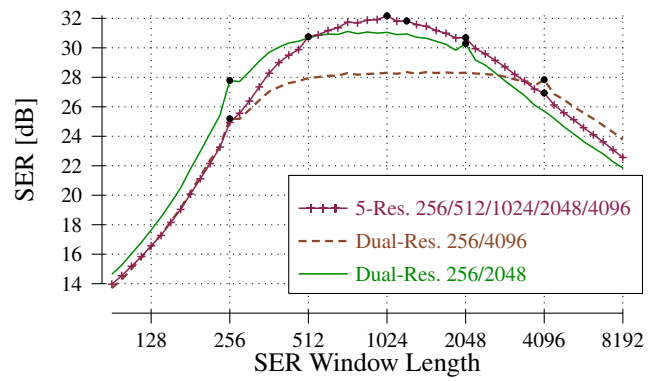


Figure 8: Average SER for 5 different window lengths compared with 256/2048 and 256/4096 dual-window length.

8. REFERENCES

- [1] V. Gnann and M. Spiertz, "Inversion of Magnitude Spectrograms with Adaptive Window Lengths," in *Proc. IEEE Int. Conference on Acoustic Speech and Signal Processing ICASSP '09*, 2009, pp. 325–328.
- [2] A. Lukin and J. Todd, "Adaptive time-frequency resolution for analysis and processing of audio," in *AES 120th Convention Paper*, 2006.
- [3] X. Zhu, G. Beauregard, and L. Wyse, "Real-Time Signal Estimation From Modified Short-Time Fourier Transform Magnitude Spectra," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 5, pp. 1645–1653, 2007.
- [4] V. Gnann and M. Spiertz, "Improving RTISI Phase Estimation With Energy Order and Phase Unwrapping," in *Proc. Int. Conference of Digital Audio Effects DAFx 10*, 2010, pp. 367–371.
- [5] D. Griffin and J. Lim, "Signal Estimation From Modified Short-Time Fourier Transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.
- [6] J. Le Roux, H. Kameoka, N. Ono, and S. Sagayama, "Fast signal reconstruction from magnitude STFT spectrogram based on spectrogram consistency," in *Proc. Int. Conference of Digital Audio Effects DAFx 10*, 2010, pp. 397–403.
- [7] J.-R. Ohm, *Multimedia Communication Technology: Representation, Transmission and Identification of Multimedia Signals*, Springer, 2004.
- [8] European Broadcasting Union, "Sound Quality Assessment Material," Tech 3253, 1988.

SOUND ANALYSIS AND SYNTHESIS ADAPTIVE IN TIME AND TWO FREQUENCY BANDS

Marco Liuni,^{*}

UMR STMS IRCAM - CNRS - UPMC
Paris, France
marco.liuni@ircam.fr

Peter Balazs,[†]

Acoustics Research Institute
Austrian Academy of Sciences
Vienna, Austria
peter.balazs@oeaw.ac.at

Axel Röbel,

UMR STMS IRCAM - CNRS - UPMC
Paris, France
axel.roebel@ircam.fr

ABSTRACT

We present an algorithm for sound analysis and resynthesis with local automatic adaptation of time-frequency resolution. There exists several algorithms allowing to adapt the analysis window depending on its time or frequency location; in what follows we propose a method which select the optimal resolution depending on both time and frequency. We consider an approach that we denote as *analysis-weighting*, from the point of view of Gabor frame theory. We analyze in particular the case of different adaptive time-varying resolutions within two complementary frequency bands; this is a typical case where perfect signal reconstruction cannot in general be achieved with fast algorithms, causing a certain error to be minimized. We provide examples of adaptive analyses of a music sound, and outline several possibilities that this work opens.

1. INTRODUCTION

Traditional analysis methods based on single sets of atomic functions offer limited possibilities concerning the variation of the resolution. Moreover, the optimal analysis parameters are often set depending on an a-priori knowledge of the signal characteristics. Analyses with a non-optimal resolution result in a blurring or sometimes even a loss of information about the original signal, which affects every kind of later treatment: visual representation, features extraction and processing among others. This motivates the research for adaptive methods, conducted at present in both the signal processing and the applied mathematics communities: they lead to the possibility of analyses whose resolution locally change according to the signal features.

We present an algorithm with local automatic adaptation of time-frequency resolution. In particular, we use *nonstationary Gabor frames* [1] of windows with compact time supports, being able to adapt the analysis window depending on its time or frequency location. For compactly supported windows fast reconstruction algorithms are possible, see [1, 2, 3]: all along the paper we will in-

dicating as *fast* a class of algorithms whose principal computational cost is due to the Fourier transform of the signal.

In the present paper we want to go a step beyond and adapt the window in time *and* frequency. This case has been detailed in [4] among others. This can be possible, and frame theory [5] would help in providing perfect reconstruction synthesis methods (if no information is lost). However, this is a typical case where the calculation of the dual frame for the signal reconstruction cannot in general be achieved with a fast algorithm: thus a choice must be done between a slow analysis/re-synthesis method guaranteeing perfect reconstruction and a fast one giving an approximation with a certain error. There are, at least, two interesting approaches to obtain fast algorithms:

- **filter bank:** the signal is first filtered with an invertible bank of P pass band filters, to obtain P different band limited signals; for each of these bands a different nonstationary Gabor frame $\{g_{k,l}^p\}$ of windows with compact time support is used, with g_k^p the time-dependent window function. The other members of the frame are time-frequency shifts of g_k^p ,

$$g_{k,l}^p = g_k^p(t - a_k^p)e^{2\pi i b_k^p l t}, \quad (1)$$

where $k, l \in \mathbb{Z}$ and a_k^p, b_k^p are the time location and frequency step associated to the p -th frame at the time index k . We will write NGF to indicate a nonstationary Gabor frame in the time case, and we will always assume to be in the painless case [6]. Each band-limited signal is perfectly reconstructed with an expansion of the analysis coefficients in the dual frame $\{\widetilde{g_{k,l}^p}\}$. Note that by this notation we denote the dual frame for a fixed p . By appropriately combining the reconstructed bands we obtain a perfect reconstruction of the original signal. An important remark is that the reconstruction at every time location is perfect as long as all the frequency coefficients within all the P analyses are used. On the other hand, for every analysis we are interested in considering only the frequency coefficients corresponding to the considered band, thus introducing a reconstruction error.

^{*} This work was supported by grants from region Ile de France

[†] This work was partially supported by the WWTF project MULAC ('Frame Multipliers: Theory and Application in Acoustics; MA07-025)

- **analysis - weighting:** the signal is first analyzed with P NGFs $\{g_{k,l}^p\}$ of windows with compact time support. Each analysis is associated to a certain frequency band, and its coefficients are weighted to match this association. We look for a reconstruction formula to minimize the reconstruction error when expanding the weighted coefficients within the union of the P individual dual frames $\cup_{p=1}^P \{g_{k,l}^p\}$.

We focus here on the second approach, in the basic case of two bands; so we split the frequency dimension into high and low frequencies, with $P = 2$. We provide the algorithm for an automatic adaptation routine: in each frequency band, the best resolution is defined through the optimization of a sparsity measure deduced from the class of Rényi entropies [7]. As for the filter bank approach, the results detailed in [8] indicate a useful solution: they give an exact upper bound of the reconstruction error when reconstructing a compactly supported and essentially band-limited signal from a certain subset of its analysis coefficients within a Gabor frame.

In the first section, the analysis-weighting method is treated with an extension of the weighted Gabor frames approach [9], which will give us a closed reconstruction formula. The second section is dedicated to the sparsity measures we use for the automatic adaptation, with an insight on how weighting techniques of the analysis coefficients can lead to measures with specific features. We then close the paper with some examples and an overview on the perspectives of our research.

2. RECONSTRUCTION FROM WEIGHTED FRAMES

Let $P \in \mathbb{N}$ and $\{g_{k,l}^p\}$ be different NGFs, $p = 1, \dots, P$, where k and l are the time and frequency location, respectively. We will consider weight functions $0 \leq w^p(\nu) \leq \infty$: for every p , they only depend on the frequency location. The idea is to smoothly set to zero the coefficients not belonging to the frequency portion which the p -th analysis has been assigned to; in this way, every analysis will just contribute to the reconstruction of the signal portion of its pertinence, so high or low frequencies respectively when $P = 2$. For each NGF $\{g_{k,l}^p\}$ we write $c_{k,l}^p = w^p(b_k^p l) \langle f, g_{k,l}^p \rangle$ to indicate the weighted analysis coefficients, and we consider the following reconstruction formula:

$$\tilde{f} = \mathcal{F}^{-1} \left(\frac{1}{p(\nu)} \mathcal{F} \left(\sum_{p=1}^P \sum_{k,l} r(p, k, l) \right) \right), \quad (2)$$

where $p(\nu) = \#\{p : w^p(\nu) \geq \epsilon\}$ and for every $\epsilon > 0$, $r(p, k, l)$ is 0 if $w^p(b_k^p l) < \epsilon$, else

$$r(p, k, l) = (w^p(b_k^p l) \langle f, g_{k,l}^p \rangle) \frac{1}{w^p(b_k^p l)} \widetilde{g_{k,l}^p}. \quad (3)$$

We see that non-zero weights cancel each other: this reconstruction formula still makes sense, as the goal is exactly to find a reconstruction as an expansion of the $c_{k,l}^p$.

We give now an interpretation of the introduced formula. If w^p is a semi-normalized sequence for each p , that is there exist constants m_p and n_p such that $0 < m_p \leq w^p(b_k^p l) \leq n_p$ and $\epsilon \leq m_p \forall p$, then $p(\nu) = p$ and the equation (2) becomes

$$\tilde{f} = \frac{1}{P} \sum_{p=1}^P \sum_{k,l} (w^p(b_k^p l) \langle f, g_{k,l}^p \rangle) \frac{1}{w^p(b_k^p l)} \widetilde{g_{k,l}^p} = f. \quad (4)$$

This is related to the concept of weighted frames detailed in [9], as in the hypothesis of semi-normalization the sequence $w^p(b_k^p l) g_{k,l}^p$ is a frame with $\frac{1}{w^p(b_k^p l)} \widetilde{g_{k,l}^p}$ as one of its dual. For weights which are not bounded from below, but still non-zero, the reconstruction still works: the sequences $w^p(b_k^p l) \cdot g_{k,l}^p$ are not frames anymore (for each p), but complete Bessel sequences (also known as upper semi-frames [10]). This reconstruction can be unstable, though.

In our case, these hypotheses are not verified, as we need to set to zero a certain subset of the coefficients within both of the analyses; thus the equation (2) will in general give an approximation of f . In section 4.2 we give an example of reconstruction following this approach, evaluating the reconstruction error; further theoretical and numerical examinations should be realized, as we are interested to find an upper bound for the error depending on:

- the signal spectral features at frequencies ν where $p(\nu) > 1$;
- the features of the w^p sequences and the $p(\nu)$ function.

A first natural choice for the weights w^p is a binary mask; first because this is the worst case in terms of reconstruction error, as we are multiplying in the frequency domain with a rectangular window before performing an inverse Fourier transform. Thus the analysis of the error with a binary masking establish a bound to the error obtained with a smoother mask. Moreover, with a binary mask the reconstruction formula takes the very simple form detailed in equation (6), allowing a direct implementation derived from the general full band algorithm. So we consider $P = 2$ and ω_c a certain cut value, then

$$w^1(\nu) = \begin{cases} 1 & \text{if } \nu \leq \omega_c \\ 0 & \text{if } \nu > \omega_c \end{cases} \quad (5)$$

and $w^2(\nu) = 1 - w^1(\nu)$. In this case $p(\nu) = 1$ for every frequency ν and the equation (2) becomes

$$\tilde{f} = \sum_{b_k^1 l \leq \omega_c} \langle f, g_{k,l}^1 \rangle \widetilde{g_{k,l}^1} + \sum_{b_k^2 l > \omega_c} \langle f, g_{k,l}^2 \rangle \widetilde{g_{k,l}^2}. \quad (6)$$

The reconstruction error in this case will in general be large at frequencies corresponding to coefficients close to the cut value ω_c ; we envisage that a way to reduce this error is to allow the w^p weights to have a smooth overlap; this results in more coefficients form different analyses contributing to the reconstruction of a same portion of signal, thus weakening their interpretation.

3. RÉNYI ENTROPY EVALUATION OF WEIGHTED SPECTROGRAMS

The representation we take into account is the spectrogram of a signal f : it is the squared modulus of the Short-Time Fourier Transform (STFT) of f with window g , which is defined by

$$\mathcal{V}_g f(u, \xi) = \int f(t) \overline{g(t-u)} e^{-2\pi i \xi t} dt, \quad (7)$$

and so the spectrogram is $\text{PS}_f(t, \omega) = |\mathcal{V}_g f(t, \omega)|^2$. Given a Gabor frame $\{g_{k,l}\}$ we obtain a sampling of the spectrogram coefficients considering $z_{k,l} = |\langle f, g_{k,l} \rangle|^2$. With an appropriate normalization, both the continuous and sampled spectrogram can be interpreted as probability densities. The idea to use Rényi entropies

as sparsity measures for time-frequency distributions has been introduced in [7]: minimizing the complexity or information of a set of time-frequency representations of a same signal is equivalent to maximizing the concentration, peakiness, and therefore the sparsity of the analysis. Thus we will consider as *best* analysis the sparsest one, according to the minimal entropy evaluation.

Given a signal f and its spectrogram PS_f , the Rényi entropy of order $\alpha > 0$, $\alpha \neq 1$ of PS_f is defined as follows

$$H_\alpha^R(PS_f) = \frac{1}{1-\alpha} \log_2 \iint_R \left(\frac{PS_f(t, \omega)}{\iint_R PS_f(t', \omega') dt' d\omega'} \right)^\alpha dt d\omega, \quad (8)$$

where $R \subseteq \mathbb{R}^2$ and we omit its indication if equality holds. Given a discrete spectrogram obtained through the Gabor frame $\{g_{k,l}\}$, we consider R as a rectangle of the time-frequency plane $R = [t_1, t_2] \times [\nu_1, \nu_2] \subseteq \mathbb{R}^2$. It identifies a sequence of points G on the sampling grid defined by the frame. As a discretization of the original continuous spectrogram, every sample $|z_{k,l}|^2$ is related to a time-frequency region of area ab , where a and b are respectively the time and frequency steps; we thus obtain the discrete Rényi entropy measure directly from (8),

$$H_\alpha^G[PS_f] = \frac{1}{1-\alpha} \log_2 \sum_{k,l \in G} \left(\frac{z_{k,l}}{\sum_{[k',l'] \in G} z_{k',l'}} \right)^\alpha + \log_2(ab). \quad (9)$$

We consider now another weight function $0 \leq w(k, l) \leq \infty$; instead of weighting the STFT coefficients $\langle f, g_{k,l} \rangle$ as we did in Section 2, we weight here the discrete spectrogram obtaining a new distribution $z_{k,l}^* = w(k, l) z_{k,l}$ which is not necessarily the spectrogram of a signal: nevertheless, by the definition of $w(k, l)$, its Rényi entropy can still be evaluated from (9). This value gives an information of the concentration of the distribution within the time-frequency area emphasized by the specific weight function: as we show in section 4.1, this can be useful for the customization of the adaptation procedure.

We will focus on discretized spectrograms with a finite number of coefficients, as dealing with digital signal processing requires to work with finite sampled signals and distributions. As α tends to one this measure converges to the Shannon entropy, which is therefore included in this larger class. General properties of Rényi entropies can be found in [11], [12] and [13]; in particular, given P a probability density, $H_\alpha(P)$ is a non increasing function of α , so $\alpha_1 < \alpha_2 \Rightarrow H_{\alpha_1}(P) \geq H_{\alpha_2}(P)$. Moreover, for every order α the Rényi entropy H_α is maximum when P is uniformly distributed, while it is minimum and equal to zero when P has a single non-zero value. As we are working with finite discrete densities we can also consider the case $\alpha = 0$ which is simply the logarithm of the number of elements in p ; as a consequence $H_0[p] \geq H_\alpha[p]$ for every admissible order α . As long as we can give an interpretation to the α parameter, this class of measures offers a largely more detailed information about the time-frequency representation of the signal.

3.1. Adaptive procedure

We choose a finite set S of admissible scaling factors, and realize different scaled version of a window g ,

$$g^s(t) = \frac{1}{\sqrt{s}} g\left(\frac{t}{s}\right), \quad (10)$$

so that the discretized temporal support of the scaled windows g^s still remains inside G for any $s \in S$. In our case, G is a rectangle with the time segment analyzed as horizontal dimension and the whole frequency lattice as vertical: at each step of our algorithm, this rectangle is shifted forward in time with a certain overlap with the previous position. By fixing an α , the sparsest local analysis is defined to be the one with minimum Rényi entropy: thus the optimization is performed on the scaling factor s , and the best window is defined consequently, with a similar approach to the one developed in [14]. With the weight functions introduced above, we are also able to limit the frequency range of the rectangle G at each time location: adaptation is thus obtained over the time dimension for each weighted spectrogram, so in our case for each frequency band enhanced. An interpolation is performed over the overlapping zones to avoid abrupt discontinuities in the tradeoff of the resolutions: in the examples given in section 4, the spectrogram segment for the entropy evaluation includes four spectrogram frames of the largest window, and the overlapping zone corresponds to three frames of the largest window. The temporal sizes of the segment and the overlap are deduced accordingly.

The time-frequency adapted analysis of the global signal is finally realized by opportunely assembling the slices of local sparsest analyses obtained with the selected windows.

3.2. Biasing spectral coefficients through the α parameter

The α parameter in equation (8) introduces a biasing on the spectral coefficients; to have a qualitative description of this biasing, we first consider a collection of simple spectrograms composed by a variable amount of large and small coefficients. We realize a vector D of length $N = 100$ generating numbers between 0 and 1 with a normal random distribution; then we consider the vectors D_M , $1 \leq M \leq N$ such that

$$D_M[k] = \begin{cases} D[k] & \text{if } k \leq M \\ \frac{D[k]}{20} & \text{if } k > M \end{cases} \quad (11)$$

and then normalize to obtain a unitary sum. We then apply Rényi entropy measures with α varying between 0 and 3: these are the values that we use to adopt for music signals. As we see from figure 1, there is a relation between the number of large coefficients M and the slope of the entropy curves for the different values of α . For $\alpha = 0$, $H_0[D_M]$ is the logarithm of the number of non-zero coefficients and it is therefore constant; when α increases, we see that densities with a small amount of large coefficients gradually decrease their entropy, faster than the almost flat vectors corresponding to larger values of M . This means that by increasing α we emphasize the difference between the entropy values of a peaky distribution and that of a nearly flat one. The sparsity measure, we consider, selects as best analysis the one with minimal entropy, so reducing α rises the probability of less peaky distributions to be chosen as sparsest: in principle, this is desirable as weaker components of the signal, such as partials, have to be taken into account in the sparsity evaluation.

The second example we consider shows that the just mentioned principle should be applied with care, as a small coefficient in a spectrogram could be determined by a partial as well as by a noise component; with an extremely small α , the best window selected could vary without a reliable relation with spectral concentration, depending on the noise level within the sound. We illustrate how noise has to be taken into account when tuning the α

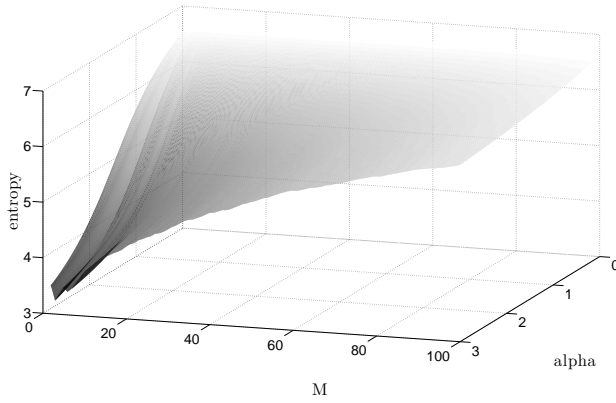


Figure 1: Rényi entropy evaluations of the D_M vectors with varying α ; the distribution becomes flatter as M increases. Therefore increasing α favors a sparse representation (see text).

parameter by means of another model of spectrogram: taking the same vector D considered previously, and two integers $1 \leq N_{part}$, $1 \leq R_{part}$, we define D_L like follows:

$$D_L[k] = \begin{cases} 1 & \text{if } k = 1 \\ \frac{D[k]}{R_{part}} & \text{if } 1 < k \leq N_{part} \\ \frac{D[k]}{R_{noise}} & \text{if } k > N_{part} \end{cases} \quad (12)$$

where $R_{noise} = \frac{R_{part}}{L}$, $L \in [\frac{1}{16}, 1]$; then we normalize to obtain a unitary sum. This vectors are a simplified model of the spectrograms of a signal whose coefficients correspond to one main peak, N_{part} partials with amplitude reduced by R_{part} and some noise whose amplitude varies, proportionally to the L parameter, from a negligible level to the one of the partials. Applying Rényi entropy measures with α varying between 0 and 3, we obtain the figure 2, which shows the impact of the noise level L on the evaluations with different values of α .

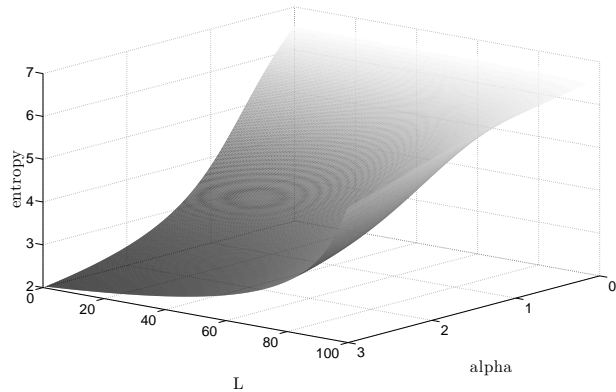


Figure 2: Rényi entropy evaluations of the D_L vectors with varying α , $N_{part} = 5$ and $R_{part} = 2$; the entropy values rise differently as L increases, depending on α : this shows that the impact of the noise level on the entropy evaluation depends on the entropy order (see text).

The increment of L corresponds to a strengthening of the noise

coefficients, causing the rise of the entropy values for any α . The key point is the observation of how they rise, depending on the α value: the convexity of the surface in figure 2 increases as α becomes larger, and it describes the impact of the noise level on the evaluation; the stronger convexity when α is around 3 denotes an higher robustness, as the noise level needs to be high to determine a significant entropy variation. Our tests show that, as a drawback, in this way we lower the sensitivity of the evaluation to the partials, and the measure keeps almost the same profile for every $R_{part} > 1$.

On the other hand, when α tends to 0 the entropy growth is almost linear in L , showing the significant impact of noise on the evaluation, as well as a finer response to the variation of the partials amplitude. As a consequence, the tuning of the α parameter has to be performed according to the desired tradeoff between the sensitivity of the measure to the weak signal components to be observed, and the robustness to noise. In our experimental experience, the value of 0.7 is appropriate for both speech and music signals.

4. ALGORITHMS AND EXAMPLES

We give here two examples of the methods described above: the first shows an application of two different weights on the spectrogram of a given sound, which determines two different choices for the optimal resolutions; the second is a reconstruction with the algorithm detailed in Section 2.

4.1. Adaptation with Different Masks

We can privilege a certain subset of the analysis coefficients to drive the adaptation routine, instead of considering them all with the same importance. For example, the adaptation within the p -th band could be determined from the coefficients laying at a certain small distance from the band central frequency.

Figures 3 and 4 are realized with an improved version of the algorithm described in [15], which allows for a weighting of the analysis coefficients which concerns only the adaptation routine, and not the analysis and re-synthesis. Thus, we obtain different adapted analyses depending on the frequency area we wish to privilege, still preserving perfect reconstruction: the sound we analyze is a music signal with a bass guitar, a drum set and a female singing voice starting from second 1.54. We use two different complementary binary masks, the first setting to zero the spectrogram coefficients corresponding to frequencies higher than 300Hz, the second doing the opposite. As we can see in Figure 3, with the first mask we obtain an analysis where the largest window is privileged; this is the best frequency resolution for the bass guitar sound, which is prominent in the considered band. The only points where shorter windows are chosen correspond to strong transients, as bass or voice attacks, where the time precision is enhanced.

With the second mask, low frequencies are ignored in the adaptation step, and as a consequence we obtain a different optimal analysis: the smallest window is generally selected, yielding an higher time resolution which is best adapted to the percussive sounds; moreover, we see that the largest window is chosen corresponding to the presence of the singing voice, whose higher harmonics belong to the considered band and determine a better frequency resolution to be privileged.

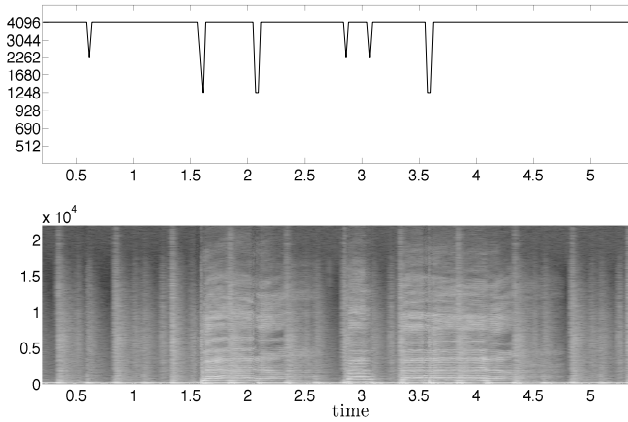


Figure 3: Adaptive analysis with a mask privileging frequencies below 300Hz, on a music signal with a bass guitar, a drum set and a female singing voice starting from second 1.54: on top, best window size chosen as a function of time; at the bottom, adapted spectrogram of the analyzed sound file.

In both cases we calculate the difference between the signal reconstructed and the original one; we use a 16 bit audio file, whose amplitude is represented in the range $[-1, 1]$ with double precision: the maximum absolute value of the differences between corresponding time samples, as well as the root mean square error over the entire signal, are both of order 10^{-16} .

4.2. Analysis-Weighting Example

We show here an example of the approximation of a signal applying the formula (6), within the analysis-weighting approach using a binary mask: as detailed in Sections 2 and 3, we analyze a signal with different stationary Gabor frames; the sound we consider is the same of the section 4.1, and the binary mask is still obtained with a cut frequency of 300Hz, while the sampling rate is 44.1kHz. We modify the coefficients of all these analyses with the mask $w^1(\nu)$, and build the NGF $\{g_{k,l}^1\}$ with resolutions adapted to the low frequencies optimizing the entropy of the masked analyses. Then we repeat this step with the mask $w^2(\nu)$ and build the NGF $\{g_{k,l}^2\}$. We finally calculate the duals of the two NGFs, which can be done in these cases with fast algorithms, and re-synthesize the two signal bands: for these examples, the reconstruction is performed with the SuperVP phase vocoder by Axel Röbel [16]. Figure 5 shows the spectrogram of the lower signal band, reconstructed with the low-frequencies adapted analysis. This spectrogram is computed with a fixed window, which is the largest one within the set considered; the choice of the best window is given as well, to give information about how the reconstruction is performed at each time. Figure 6 is obtained in the same way, considering the upper band reconstruction. The approximation of the original sound is then given by the sum of the two bands.

The reconstruction error we obtain is higher than the one in the previous examples: the maximum absolute value of the samples differences is 0.0568, while the root mean square error is 0.0099. With the choice of a binary mask, the only way to reduce the error is to set the cut frequency in a range where the signal energy is

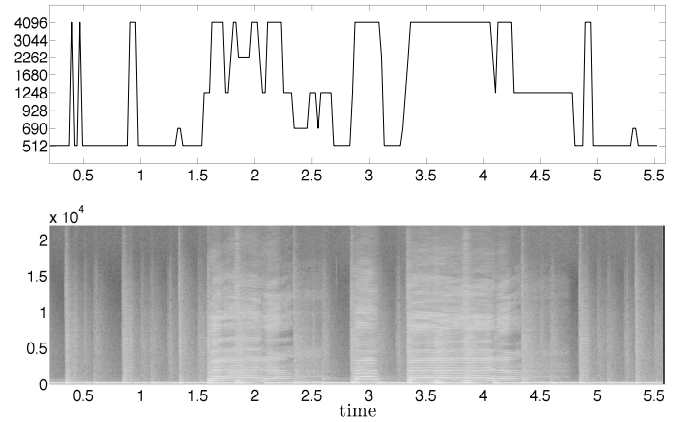


Figure 4: Adaptive analysis with a mask privileging frequencies above 300Hz, on a music signal with a bass guitar, a drum set and a female singing voice starting from second 1.54: on top, best window size chosen as a function of time; at the bottom, adapted spectrogram of the analyzed sound file.

low: unfortunately, music signals generally do not have large low-energy bands; moreover, the interest of our method relies in the possibility for the cut frequency to be variable, in order to freely select the adaptation criterion.

Figure 7 shows the spectrogram of the difference between the original sound and the reconstructed one, and we see that the spectral content of the error is concentrated at the cut frequency. The alteration introduced has negligible perceptual effects, so that the original signal and the reconstruction are hard to be distinguished: this aspect needs to be quantified; when dealing with the approximation of music signals, the objective error measures do not give any information about the perceptual meaning of the error. The accuracy of a method has thus to be evaluated by means of measures taking into account the human auditory system as well as listening tests.

Another element to consider is the overlap between the weight functions introduced in section 2: if we allow them for an overlap over a sufficiently large frequency band, we envisage that the error would be reduced. The sense of this point can be clarified considering the causes of the reconstruction error: windows with compact time support cannot have a compactly supported Fourier transform; from the analysis point of view, this means that a spectrogram coefficient affects the signal reconstruction among the whole frequency dimension. We can limit such an influence with a choice of well-localized time-frequency atoms: even if their frequency support is not compact, they have a fast decay outside a certain region. If we cut with a binary mask outside a certain band, the reconstruction error comes mainly from the fact that we are setting to zero the contribution of atoms whose Fourier transforms spread into the band of interest: if the atoms are well-localized, only a few of them actually have an impact.

Formula (2) gives an ideal reference: if the overlap is the entire frequency dimension, weights are non-zero, thus we have a perfect reconstruction from the weighted coefficients. When some weights are zero and weight functions do overlap, the normalization factor in the formula (2) is greater than one in the overlapping

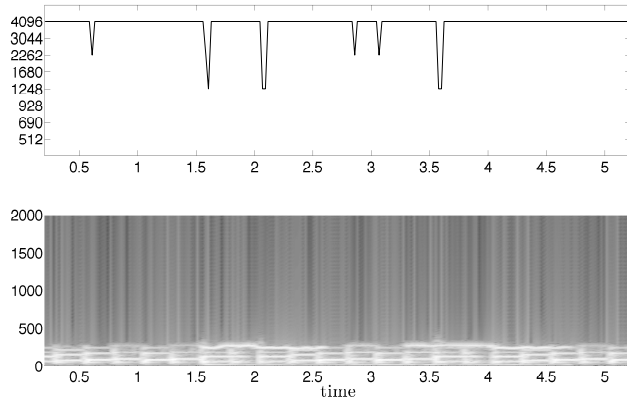


Figure 5: Low-frequencies reconstruction from the masked adapted analysis of a music signal with a bass guitar, a drum set and a female singing voice starting from second 1.54: on top, best window size chosen as a function of time. At the bottom, spectrogram of the analyzed band with a 4096 samples Hamming window, 3072 samples overlap and 4096 frequency points; the frequency axis is bounded to 2kHz to focus on the reconstructed region.

frequency interval. This reduces the impact of the errors coming from individual re-syntheses: on the other hand, the fact of summing them all imposes a limit to the achievable global error reduction.

A further improvement of this formula is to put different weights at the denominator in (4), with an effective amplification or reduction of the contributions coming from individual coefficients. To keep the perfect reconstruction valid in the case of semi-normalized norms, a possibility is to obtain the different weights as a function of the analysis weights depending also on the overlap.

5. CONCLUSIONS AND PERSPECTIVES

We have sketched the first steps of a promising research project about the local automatic adaptation of time-frequency sound representations: a first question which arises is how to display a representation of the signal such the one described; there are two possibilities involving weighted means of the coefficients at a certain time-frequency location:

- $d_{k,l} = \frac{1}{\sum_p w^p} \cdot \sum_p c_{k,l}^p$, displaying $|d_{k,l}|$, or
- $d_{k,l}^{(A)} = \frac{1}{\sum_p w^p} \cdot \sqrt{\sum_p |c_{k,l}^p|^2}$.

In a previously proposed method [15] the algorithm keeps the original coefficients in memory; with this approach, we can use the reconstruction scheme mentioned in (13). A further new question would be how to reconstruct the signal from an expansion of the $d_{k,l}$ or $d_{k,l}^{(A)}$ coefficients. Straightforward numerical examples could give some numerical insights.

If $d_{k,l}^{(A)}$ is used, we also have to address the problem of the phase. This approach is useful when dealing with spectrogram transformations where the phase information is lost, as with re-assigned spectrogram or spectral cepstrum. We could either use an iterative approach, like the one described in [17] adapted to frame

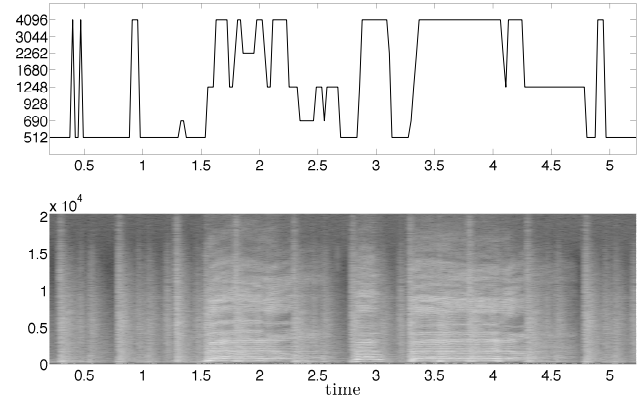


Figure 6: High-frequencies reconstruction from the masked adapted analysis of a music signal with a bass guitar, a drum set and a female singing voice starting from second 1.54: on top, best window size chosen as a function of time; at the bottom, spectrogram of the analyzed band with a 4096 samples Hamming window, 3072 samples overlap and 4096 frequency points.

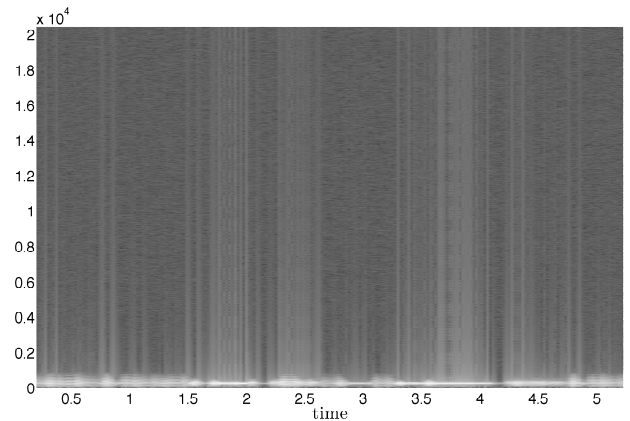


Figure 7: Spectrogram of the reconstruction error given by the described method on a music signal with a bass guitar, a drum set and a female singing voice starting from second 1.54; spectrogram obtained with a 4096 samples Hamming window, 3072 samples overlap and 4096 frequency points.

theory, or use a system with a high redundancy (see [18]).

From a computational point of view, we are interested in limiting the size of the signal for the direct and inverse Fourier transforms in (2), as this will largely improve the efficiency of the algorithm. A different form of the formula (2) in this sense is

$$\tilde{f} = \sum_{p,k,l} c_{k,l}^p \mathcal{F}^{-1} \left(\frac{1}{p(\nu)} \mathcal{F} \left(\frac{\widetilde{g_{k,l}^p}}{w^p(b_k^p l)} \right) \right) \quad (13)$$

whose properties have to be further investigated.

Later we would also investigate the properties of time-variant filters by multiplying these new sets of coefficients, resulting in new kinds of frame multipliers [19]. Using an optimized way to analyze acoustical signal, will, therefore, also lead to a better control of such adaptive filters.

6. REFERENCES

- [1] F. Jaillet, P. Balazs, M. Dörfler, and N. Engelpützeder, “Non-stationary Gabor Frames,” in *Proc. of SAMPTA’09*, Marseille, France, May 18-22, 2009.
- [2] P. Balazs, M. Dörfler, N. Holighaus, F. Jaillet, and G. Velasco, “Theory, Implementation and Applications of Nonstationary Gabor Frames,” submitted, http://www.univie.ac.at/nonstatgab/pdf_files/badohojave11_04042011.pdf, 2011.
- [3] P. L. Søndergaard, B. Torrèsani, and P. Balazs, “The Linear Time Frequency Analysis Toolbox,” <http://www.univie.ac.at/nuhag-php/lftfat/toolboxref.pdf>.
- [4] Monika Dörfler, “Quilted frames - a new concept for adaptive representation,” *Advances in Applied Mathematics*, to appear, 2010, <http://arxiv.org/pdf/0912.2363>.
- [5] O. Christensen, Ed., *An Introduction To Frames And Riesz Bases*, Birkhäuser, Boston, Massachusetts, USA, 2003.
- [6] I. Daubechies A. Grossmann Y. Meyer, “Painless nonorthogonal expansions,” *J. Math. Phys.*, vol. 27, pp. 1271–1283, May 1986.
- [7] R.G. Baraniuk P. Flandrin A.J.E.M. Janssen O.J.J. Michel, “Measuring Time-Frequency Information Content Using the Rényi Entropies,” *IEEE Trans. Info. Theory*, vol. 47, no. 4, pp. 1391–1409, May 2001.
- [8] E. Matusiak Y. C. Eldar, “Sub-Nyquist sampling of short pulses: Part i,” <http://arxiv.org/abs/1010.3132v1>.
- [9] P. Balazs, J.-P. Antoine, and A. Griboś, “Weighted and controlled frames: mutual relationships and first numerical properties,” *Int. J. Wav. Mult. Info. Proc.*, vol. 8, no. 1, pp. 109–132, 2010.
- [10] J.-P. Antoine and P. Balazs, “Frames and semi-frames,” to appear in *Journal of Physics A: Mathematical and Theoretical*. <http://arxiv.org/pdf/1101.2859v2>.
- [11] A. Rényi, “On Measures of Entropy and Information,” in *Proc. Fourth Berkeley Symp. on Math. Statist. and Prob.*, Berkeley, California, June 20-30, 1961, pp. 547–561.
- [12] F. Schlögl C. Beck, Ed., *Thermodynamics of chaotic systems*, Cambridge University Press, Cambridge, Massachusetts, USA, 1993.
- [13] K. Zyczkowski, “Rényi Extrapolation of Shannon Entropy,” *Open Systems & Information Dynamics*, vol. 10, no. 3, pp. 297–310, Sept. 2003.
- [14] F. Jaillet and B. Torrèsani, “Time-frequency jigsaw puzzle: adaptive and multilayered Gabor expansions,” *International Journal for Wavelets and Multiresolution Information Processing*, vol. 1, no. 5, pp. 1–23, 2007.
- [15] M. Liuni A. Röbel M. Romito X. Rodet, “A reduced multiple Gabor frame for local time adaptation of the spectrogram,” in *Proc. of DAFX10*, Graz, Austria, September 6-10, 2010, pp. 338 – 343.
- [16] Axel Röbel, “SuperVP,” <http://anasynth.ircam.fr/home/software/supervp>.
- [17] D.W. Griffin J.S. Lim, “Signal Estimation from Modified Short-Time Fourier Transform,” *IEEE Trans. Acoust. Speech Signal Process.*, vol. 32, no. 2, pp. 236–242, Apr. 1984.
- [18] Radu Balan, Pete Casazza, and Dan Edidin, “On signal reconstruction without phase,” *Appl. Comput. Harmon. Anal.*, vol. 20, no. 3, pp. 345–356, 2006.
- [19] P. Balazs, “Basic definition and properties of Bessel multipliers,” *Journal of Mathematical Analysis and Applications*, vol. 325, no. 1, pp. 571–585, January 2007.

EXPONENTIAL FREQUENCY MODULATION BANDWIDTH CRITERION FOR VIRTUAL ANALOG APPLICATIONS

Joseph Timoney, Victor Lazzarini,
Sound and Music Technology Research
Group, NUI Maynooth
Maynooth, Ireland
jtoney@cs.nuim.ie
victor.lazzarini@nuim.ie

ABSTRACT

Cross modulation or Exponential FM is a sound synthesis technique associated with modular analog subtractive synthesizers. It differs from the more well-known linear FM synthesis technique in that the modulation is an exponential function of the control voltage. Its spectrum shape is more complex, thus giving it a larger bandwidth with respect to the modulation depth. Thus, the prevention of aliasing distortion requires different conditions than Carson's rule as used with linear FM. A suitable equation will be presented in this paper.

1. INTRODUCTION

Research into virtual implementations of the structures of analog subtractive synthesizers has been a popular topic for the last few years. There have been a number of algorithms proposed to create bandlimited versions of the classic analog oscillators [1]. Alongside this, there have been a number of papers that derive models for the Voltage controlled filters associated with particular analog synthesizers. The designs for these have been based on an explicit circuit analysis, [2] for example, or those that create a version of the original using standard digital filter elements [3]. However, other elements of subtractive synthesis systems have not received such in-depth treatment such as the Attack-Decay-Sustain-Release (ADSR) envelope generators, Filter FM effects, and other oscillator modulation configurations. Although on modern analog synthesizers linear Frequency modulation (FM) between oscillators is sometimes a feature this was not always that case. In fact, Linear Frequency Modulation is really associated with digital synthesis and was viewed as the synthesis technology that overtook subtractive synthesis in the 1980s [4]. Linear Frequency modulation was unavailable in early modular subtractive synthesis systems for two primary reasons: Firstly, it was difficult to implement because of the Volts/Octave control voltage concept on which the elements of these synthesizers were interconnected. This system creates a non-linear relationship between any change in signal voltage and the pitch [5]. Thus, to force it to behave in a linear manner was difficult. Secondly, the tuning instabilities associated with early analog synthesizers because of component drift and ambient temperature fluctuations would have resulted in inconsistent generation of stable linear FM signals [6]. This is particularly important in the case when the linear FM signal is desired to be harmonic; to achieve this, the frequencies of the carrier and modulator signals must strictly be in an integer relationship. Any deviation from this can seriously affect the perception of the sound.

The nonlinear Volts/Octave control voltage relationship of these analog synthesizers meant that modulation of one oscillator by another actually produced an Exponential FM waveform. One notable feature of this technique was that when the modulation depth was varied dynamically, a pitch shifting of the sound was perceived. This issue meant that for musicians Exponential FM was most often used to produce special effect sounds that were clangorous rather than for melodic lines. An excellent treatment of the theory of Exponential FM was written by [7]. This paper provided a description of the signal and its spectrum for sine-wave modulators. It also offered a configuration of analog modules that introduced a pitch correction factor that could be used to produce a harmonic version of Exponential FM. However, the work in [7] was written for implementation on an analog synthesizer system and thus assumed an infinite output bandwidth. This is not the case for digital implementations and algorithm designers always have to be aware of limitations on signal bandwidth imposed by the sampling frequency so as to minimise aliasing distortion. Therefore, this paper will examine the implementation of Exponential FM from a digital perspective. It will examine the spectra produced by the Exponential FM system in an effort to produce a guideline for its digital implementation. Section 2 will introduce the theory behind Exponential FM and will provide the spectral bandwidth analysis. Section 3 will contain the conclusions.

2. THEORY OF EXPONENTIAL FM

First of all, the Volts/Octave control signal representation in analog synthesizers means that the pitch of a note doubles as the control voltage doubles [5]. Thus, the relationship can be expressed

$$f \propto 2^V \quad (1)$$

where f is the note pitch and V is the control voltage.

For example, if 2 volts produces a pitch of 110Hz, then 3 volts will produce a note an octave higher of 220Hz.

Next, although the well-known equation for linear FM synthesis is actually phase modulation, on a modular synthesizer system Exponential FM is implemented as a true frequency modulation. The modulator signal voltage is interpreted as frequency variation that is used to define the control voltage associated with frequency of the carrier. The modulator is thus an instantaneous frequency signal that for exponential FM is defined as in [7] to be

$$\dot{\theta}(t) = f_c 2^{V(t)} \quad (2)$$

where f_c and f_m are the Carrier and Modulation frequencies in hertz respectively, and $V(t)$ represents the modulating signal. Note that the dot on the term on the left hand side indicates that it is a differential (of the phase).

The Modulating signal can be written as the combination of a DC term, V_0 and a time-varying quantity, assumed to be a cosine here, of amplitude V_m , which can also be termed as the Modulation Depth [7],

$$V(t) = V_0 + V_m \cos(2\pi f_m t) \quad (3)$$

Substituting (3) into (2) and using logarithms to write the power term

$$\dot{\theta}(t) = f_c e^{(V_0 + V_m \cos(2\pi f_m t)) \ln(2)} \quad (4)$$

which can be rewritten

$$\dot{\theta}(t) = f_{ce} e^{(V_m \cos(2\pi f_m t)) \ln(2)} \quad (5)$$

with

$$f_{ce} = f_c e^{V_0 \ln(2)} \quad (6)$$

To convert the instantaneous frequency signal into a phase the exponential term in (5) must be integrated

$$\theta(t) = \int \dot{\theta}(t) \quad (7)$$

However, it is not possible to integrate the exponential term in (5) directly and instead it must be expanded as set of Modified Bessel functions [7]

$$e^{V_m \cos(2\pi f_m t) \ln(2)} = I_0(V_m \ln(2)) + 2 \sum_{k=1}^{\infty} I_k(V_m \ln(2)) \cos(2\pi k f_m t) \quad (8)$$

Substituting (8) back into (5)

$$\dot{\theta}(t) = I_0(V_m \ln(2)) f_{ce} + \left(2 \sum_{k=1}^{\infty} I_k(V_m \ln(2)) \cos(2\pi k f_m t) \right) f_{ce} \quad (9)$$

Integrating to obtain the phase as shown by (7) and assuming a sinusoidal carrier will produce the time domain signal [7]

$$y(t) = \sin(\theta(t)) = \sin \left(2\pi I_0(V_m \ln(2)) f_{ce} t + \left(\sum_{k=1}^{\infty} \frac{2f_c}{kf_m} I_k(V_m \ln(2)) \sin(2\pi k f_m t) \right) \right) \quad (10)$$

The carrier frequency of (11) can be written as

$$f_E = I_0(V_m \ln(2)) f_{ce} \quad (11)$$

and the frequency deviation of each term is

$$D_k = \frac{2f_c}{kf_m} I_k(V_m \ln(2)) \quad (12)$$

The expression in (10) is a multi-component complex FM signal.

2.1. Carrier Frequency Analysis

From (6) and (10) it can be seen that the final carrier frequency of the Exponential FM signal is a function of the exponent of the DC term V_0 and the value of the zeroth modified Bessel function that has the Modulation Depth in its argument. This is different to the linear FM case where the carrier frequency is independent of the modulation. This relationship should be taken into account for the digital version of Exponential FM. For example, assuming for convenience that the DC term V_0 is zero it is possible to show graphically how the carrier frequency increases with increasing modulation amplitude. This is illustrated in Figure 1 where the input carrier frequency f_c is plotted along the x-axis, the Modulation Depth V_m on the y-axis, and the actual carrier frequency given by (11) on the z-axis. The maximum possible value of the input carrier was assumed to be 8372Hz, corresponding to midi-note #120.

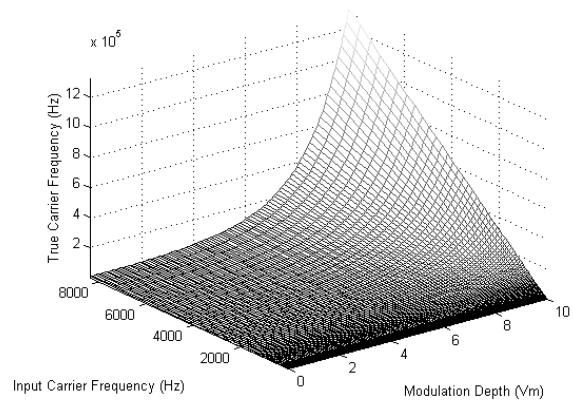


Figure 1: The relationship between the input carrier frequency, the modulation depth and the actual carrier frequency.

From Figure 1 it can be seen the actual carrier frequency increases quite rapidly as the Modulation Depth V_m reaches values of 8 or more. This illustrates the difference between linear FM and Exponential FM well. It also hints at the problems that can occur with the digital implementation of Exponential FM and warns that care must be taken when setting a sampling frequency for any implementation so that it is commensurate with the width of the Modulation Depth control. Lastly, looking at (11) it can be seen that including a DC term (V_0) in the modulating signal adds further complications in that it can raise the carrier frequency significantly. For example, for $V_0=5$ the carrier frequency will be scaled by a factor of 7.17.

2.2. Computing the Spectrum of Exponential FM

To obtain the spectrum of the Exponential FM signal in (10) there are a number of possible approaches. These can be numerical, analytical, or a hybrid of the two. Note though what is more useful here is the spectrum envelope rather than the actual spectrum itself. When attempting to compute the bandwidth it is much easier to work with the envelope because any gaps that exist between the partials in the signal that can disrupt an auto-

mated spectral analysis process to find a significant low energy spectral region.

The most obvious numerical technique to use is the Fast Fourier Transform (FFT). This is readily available in most software environments. However, this does not produce the envelope itself and further processing is required for this. Methods to achieve this include autoregressive analysis or Cepstral techniques.

Another approach is to employ the semi-analytic method of [8] that expresses the frequency modulation itself as a piecewise linear function. The total spectrum of the modulated output is the summation of the spectra for each piecewise-linear modulated part of the entire waveform. In essence this models the modulated waveform as a succession of linear Chirp signals, and thus the spectrum is the combination of the spectra for these chirps. In [8] it is proposed that the spectrum of the chirp signal is obtained using a numerical evaluation of the Fresnel equations. A faster estimate can be obtained using a Stationary Phase Approximation (SPA) and it also does not have any associated numerical integration issues [9]. However, efforts to apply this technique were not successful. It resulted in an approximate spectrum that had a blocky appearance which was not a particularly good match to the FFT based spectrum. It neither captured the true height of the various spectral components or the complete width of the spectrum.

Aside from the quasi-static approach for spectral approximation that is allowable under very particular conditions [10], the most general analytical approach is to expand the modulation term of (10) using Bessel functions [10]. Rewriting (10) by substituting (11) and (12)

$$y(t) = \sin\left(2\pi f_c t + \left(\sum_{k=1}^{\infty} D_k \sin(2\pi k f_m t)\right)\right) \quad (13)$$

It can be expanded as [10]

$$\sin\left(\omega n + \sum_{i=1}^k D_i \sin(\omega_i n + \phi_i)\right) = \sum_{k_1} \dots \sum_{k_k} \left(\prod_{i=1}^k J_{k_i}(D_i)\right) \sin\left(\omega n + \left(\sum_{i=1}^k k_i (\omega_i n + \phi_i)\right)\right) \quad (14)$$

An example of using (14) to produce the magnitude spectrum is given in Figure 2 for $f_c=440\text{Hz}$ and $f_m=44\text{Hz}$ and $V_m=2$. In the figure the reflection of the negative frequencies back into the positive frequency region was not carried out as these always introduce spectral zeros for this particular signal case, while what is required is a smooth spectral envelope. It can be seen in Figure 2 that the envelope has a number of resonant peaks that become wider with respect to increasing frequency. The most significant component is below about half the carrier frequency. Although the higher frequency components are smaller than the carrier the spectrum does not have a true lowpass shape, but rather curves upwards at the last resonant peak.

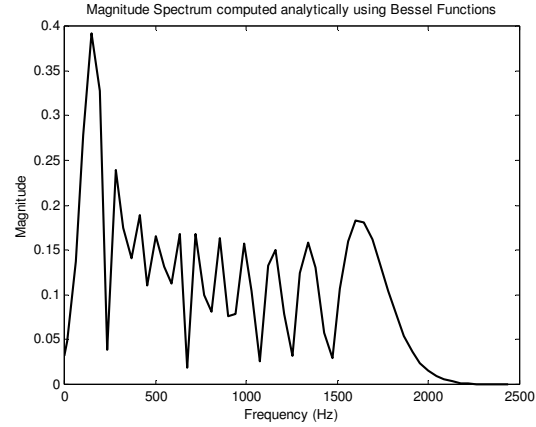


Figure 2: Magnitude spectrum of an Exponential FM signal using a Bessel Function Analysis.

There are two difficulties with the Bessel function based spectral analysis. Firstly, given that (14) is a complicated expression involving product terms, it is only possible to evaluate it numerically. This means that it cannot provide an intuitive compact expression. Secondly, the complexity of the evaluation of the equation grows exponentially with each additional sinusoidal term in the modulation signal. Some savings can be made by eliminating the evaluation of low amplitude Bessel function terms from the computation, and using programming optimisations to avoid the nested loop implementation.

2.3. Spectral Bandwidth Evaluation

The spectral bandwidth such that aliasing components would be of sufficiently low magnitude was defined to be point at which the spectrum was 80dB below the peak value. This was a reasonably strict criteria and much stronger than Carson's rule [10]. The intention was to express the Bandwidth as a function of the carrier frequency and the Modulation Depth.

First, using (14) the spectra of Exponential FM signals were computed for different values of carrier frequency and with fixed values for the modulation frequency and modulation depth. It was found that under these conditions the spectra were simply translated in relation to the carrier meaning shape invariant to the carrier frequency.

Next, spectra were again generated with the modulation frequency was expressed as a ratio of the carrier frequency from 0.1 up to 1 for a fixed value of Modulation Depth, and the bandwidth measured by an automated analysis in each case. This was repeated for other values of Modulation Depth. Figure 3 shows a plot of the results for values of Modulation Depth $V_m=1, 2$ and 3. The Bandwidth relative to the carrier frequency is shown on the y-axis as a multiple of the carrier frequency. From the plot it can be seen that the relationship between the Modulation frequency and relative Bandwidth is almost linear for all values of V_m . Thus, a simple linear fit can be made to characterize the relationship.

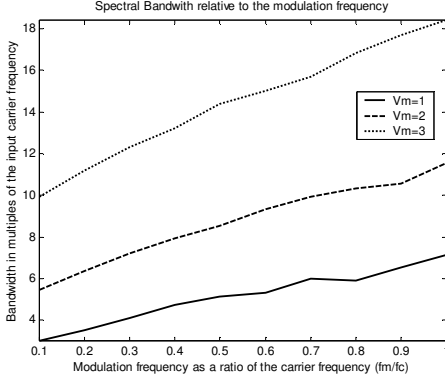


Figure 3: Spectral Bandwidth shown as a function of the Modulation frequency expressed as a ratio of the carrier frequency for three different values of Modulation Depth.

For the three different values of V_m in the figure the coefficients calculated are given in Table 1

V_m	p_0	p_1
1	2.771	4.3030
2	5.168	6.4606
3	9.3841	9.2485

Table 1: Coefficients linear fit to data in Figure 4 for different values of Modulation Depth.

The equation for the relative Bandwidth (to be multiplied by f_c for the value in Hz) was

$$BW(f_c, f_m, V_m) = p_0 + p_1(f_m/f_c) \quad (15)$$

To create a more general expression that also includes V_m on the right hand side of (15) the values in Table I can be examined. It can be seen that as V_m increases the value for p_1 increases approximately by a factor V_m . Similarly, the value for p_0 increases by about $V_m - 1$ to the power of 2. Incorporating this along with possible scaling of f_c by the DC term V_0 , (15) gives the final expression for the -80dB bandwidth in Hertz

$$BW_{-80dBHz}(f_c, f_m) = f_c e^{V_0 \ln(2)} 2^{V_m-1} p_{01} + (p_{11} + V_m) f_m \quad (16)$$

where $p_{01} = 2.771$ and $p_{11} = 4.3030$.

Two examples are given in Figure 4 to illustrate the validity of this expression. These are shown in Figure 4. In the upper panel the values to generate the Exponential FM signal were $f_c=100\text{Hz}$ and $f_m=250\text{Hz}$, $V_0=1$ and $V_m=4$. Its actual spectrum was computed using a Hanning windowed FFT and is displayed using a dB scale. The equation in (16) was used to find the bandwidth in Hz. This is plotted using the dashed vertical line in the figure. It clearly marks a point close to -80dB in front of the primary spectral region. In the lower panel the values were $f_c=10\text{Hz}$ and $f_m=40\text{Hz}$, $V_0=0$ and $V_m=7$. Again, the dB magnitude FFT was found and the bandwidth was plotted using a dashed vertical line. In this case it actually estimates the -80dB to be at a greater location in frequency. However, this error is acceptable as it is an overestimation rather than an underestimation.

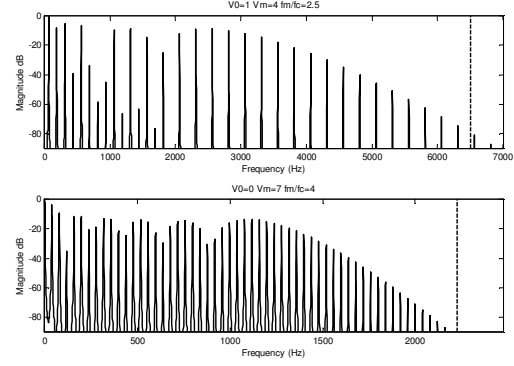


Figure 4: Example plots of FFT spectra of Exponential FM signals with the -80dB Bandwidth in Hz highlighted using a dashed line in both panels. Upper panel parameters were $f_c=100\text{Hz}$ and $f_m=250\text{Hz}$, $V_0=1$ and $V_m=4$ and for the lower panel were $f_c=10\text{Hz}$ and $f_m=40\text{Hz}$, $V_0=0$ and $V_m=7$.

3. CONCLUSIONS

This paper has presented an expression to evaluate the -80dB bandwidth of a cosine modulated Exponential FM signal. Examples were given to demonstrate its effectiveness. It should be a useful formula for low-aliasing digital implementations of Exponential FM. Future work aims to produce an exact analytical expression for the spectrum of the Exponential FM signal. It will also investigate bandwidth criteria for cases when the modulation is not sinusoidal, such as for sawtooth and square waves.

4. REFERENCES

- [1] Välimäki, V., and A. Huovilainen 2007, 'Antialiasing oscillators in subtractive synthesis,' *IEEE Signal Processing Magazine*, 24(2): 116–125.
- [2] M. Civolani and F. Fontana, 'A nonlinear digital model of the EMS VCS3 voltage controlled filter', *Proc. Dafx 2008*, Espoo, Finland, Sept. 2008.
- [3] V. Välimäki and A. Huovilainen, 'Oscillator and filter algorithms for virtual analog synthesis' *Computer Music Journal*, vol. 30, no. 2, pp. 19-31, 2006.
- [4] J. Chowning and D. Bristow, *FM theory and applications – By Musicians for musicians*, Yamaha, Tokyo, Japan, 1986.
- [5] H. Chamberlain, H., *Musical Applications of Microprocessors*, Hayden Books, Indianapolis, Indiana, USA, 1987.
- [6] M. Russ, *Sound synthesis and sampling*, Focal press, Elsevier, Oxford, UK, 2009.
- [7] B. Hutchins, 'The frequency modulation spectrum of an exponential voltage-controlled oscillator,' *Jnl. Of Audio Eng. Soc.*, Vol. 23(3), April 1975, pp. 200-206.
- [8] J. Martin, A. Holt and G. Salkeld, 'Novel analytic technique for obtaining the spectrum associated with piecewise linear FM,' *Proc. IEE*, Vol. 122(7), Jul. 1975, pp. 710-712.
- [9] E. Chassande-motin and P. Flandrin, 'On the stationary phase approximation of chirp spectra', *Proc. IEEE TITS 1998*, Pittsburg, PA, USA, Oct. 1998.
- [10] H. Rowe, *Signals and noise in communications systems*, Van Nostrand, NJ, USA, 1965.

TOWARDS ONTOLOGICAL REPRESENTATIONS OF DIGITAL AUDIO EFFECTS

Thomas Wilmering, György Fazekas and Mark B. Sandler

Centre for Digital Music
Department of Electronic Engineering
Queen Mary, University of London
Mile End Road, London E1 4NS, UK

{thomas.wilmering, gyorgy.fazekas, mark.sandler}@eecs.qmul.ac.uk

ABSTRACT

In this paper we discuss the development of ontological representations of digital audio effects and provide a framework for the description of digital audio effects and audio effect transformations. After a brief account on our current research in the field of high-level semantics for music production using Semantic Web technologies, we detail how an Audio Effects Ontology can be used within the context of intelligent music production tools, as well as for musicological purposes. Furthermore, we discuss problems in the design of such an ontology arising from discipline-specific classifications, such as the need for encoding different taxonomical systems based on, for instance, implementation techniques or perceptual attributes of audio effects. Finally, we show how information about audio effect transformations is represented using Semantic Web technologies, the Resource Description framework (RDF) and retrieved using the SPARQL query language.

1. INTRODUCTION

Research in musical applications of metadata have produced a rich literature in recent years. This includes use cases for creative music production as well as information retrieval. Brazil [1] for example exploits *cue points* or markers, which can be found in modern audio file formats, for browsing music collections. Gomez [2] introduces the use of metadata for content-based audio processing, while Pampalk [3] uses metadata for organising sample libraries. The use of content-derived information for creating adaptive audio effects was described by Verfaillie et al. [4]. Previously, the authors also exploited metadata in creative applications including navigation of recording projects using segmentation [5], and more recently we introduced a new class of audio effects where the use of standardised metadata is deeply embedded into the process of applying audio effects [6].

This system enables the prediction of changes in metadata when simple effects are applied to an audio signal, and also provides means for tracking the application of audio effects in the music production workflow. During the development of this system we identified the need for closely linked information describing data flow in different system components, and the need for a common way of representing information about audio features, as well as the characteristics and parameters of audio effects. These requirements point to the need for using a common knowledge representation framework for inter-disciplinary classification of audio effects. While such a classification has been proposed previously [7], standardised schema were not employed to represent this knowledge.

We opt for adopting Semantic Web [8] technologies for our purposes, in recognition that they provide a uniform way of encoding and linking information, governed by shared ontology schema, as well as support high-level logical reasoning based on Description Logics [9]. In particular, we use Semantic Web ontologies which provide for an *explicit specification of a conceptualisation* [10]. Data expressed using our ontologies support a wide range of use cases in creative music production, as well as exchanging accurate production data between tools, and sharing data for example with an artist community on the Semantic Web. In our research we exploit previous work on developing such ontologies and applications (see [11] for details) and develop an ontology based representation of audio effects within a common ontological framework for representing music related information and in particular studio production. This distinguishes our work from previous research where the use of metadata was only considered in isolation.

In the rest of this paper, after a brief review of Semantic Web technologies, we give an overview of the Music [12] and Studio Ontologies¹. We discuss different approaches of developing an audio effects ontology, and demonstrate a use case of retrieving detailed information using metadata describing a musical mixture.

2. SEMANTIC WEB TECHNOLOGIES

Semantic Web Technologies refer to a set of web standards for creating a "Web of Data". The purpose of the Semantic Web, as an extension to the World Wide Web, is to allow for the development of applications that are capable of exploiting the meaning of knowledge represented in Web pages. At its core, Uniform Resource Identifiers (URI) are assigned to each resource including ontological concepts and relationships, while the Resource Description Framework (RDF) defines the standard for the formal description of these resources. The RDF data model expresses statements about resources as sets of *triples* in the form of *subject, predicate, object*. The model can be seen as directed graphs, where nodes represent the subjects and objects of statements while, arcs correspond to predicates². Graph nodes may either be named by URIs, literals (e.g. strings or numbers), or *blank nodes*³. The example in listing 1 shows how an audio effect implementation is described in RDF. The triple `:rdfx_delay fx:implementation_of fx:Echo` identifies the subject as an implementation of an echo effect.⁴ The following lines describe further attributes of the im-

¹ Available from: <http://motools.sourceforge.net/>

² Predicate describes relationships between subjects and objects

³ Nodes may remain unlabelled for brevity.

⁴ Namespace prefixes such as *fx* correspond to ontologies.

plementation, such as plugin type, name, and rights-related data using DCMI Metadata Terms⁵. This representation makes use of the RDF Schema Language (RDFS) for describing properties and classes of RDF resources, and our ontology developed in the Web Ontology Language (OWL)⁶ for further refinements providing unambiguous representation of data and data relationships.

```
:rdfx_delay fx:implementation_of fx:Echo;
fx:plugin_type fx:Rdfx;
dc:description "Feedback Delay";
dc:rights "Copyright (c) 2010-2011 QMUL";
dc:title "RDFx_Delay_1";
foaf:maker [ a foaf:Agent ;
            foaf:name "Thomas Wilmering" ] .
```

Listing 1: RDF data describing an audio effect implementation.

3. THE MUSIC AND STUDIO ONTOLOGY FRAMEWORKS

Semantic Web technologies play an increasingly large role in information management for multimedia applications. Detailed ontologies dealing with various aspects of music related information have already been published for music information retrieval (MIR) use cases. The Music Ontology in particular defines concepts and relationships for this domain, on top of two fundamental ontologies: Event⁷ and Timeline⁸ for expressing time-based events. This can be used to represent concepts such as a *recording session*, but also, with further ontological support, concepts such as note onsets, or the extent of a key segment in an audio file [11]. Our research extends this modular framework with ontologies for metadata-generation and usage in the music production environment.

The Studio Ontology (STUDIO) describes recording studio concepts and provides a framework for collecting metadata in audio production. It provides extensions containing specialised terms, including an ontology of multitrack recording⁹, which enables linking elements of multitrack production tools, for example audio clips and tracks, to more general Music Ontology data [13]. The Audio Effects Ontology (FXO) is developed within this framework. The application of audio effects is an integral part of contemporary music production, therefore it was included in the core Studio Ontology. However, the need for accommodating different view points in audio effect classification gave rise to its modularisation. For example, classification based on implementation techniques or perceptual attributes require different ontology modules. The problems arising when attempting to unify these different approaches have been discussed in [7], and an interdisciplinary classification system was proposed. We show that Semantic Web ontologies, as opposed to classic taxonomies, provide a way to describe audio effects not only for classification purposes, but also for the creation and retrieval of detailed metadata about a music production.

4. THE AUDIO EFFECTS ONTOLOGY

This section deals with metadata requirements for audio effects and the development of the Audio Effects Ontology consisting of

three parts, one describing effect transformations, another providing a DAFX taxonomy and a third defining provenance terms. The motivation behind the development of the Audio Effects Ontology is to describe the domain of audio effects taking into account different view points. A first step towards this is the development of several ontologies each covering the perspective of a particular discipline, such as composition, post-production/audio engineering or effects development. In this chapter we focus on classification strategies for audio effects in ontological representations.

4.1. Effect Classification

As mentioned in §3 there are different schemata by which audio effects can be classified. For instance, we can group audio effects by the perceptual attributes that are mainly modified by their application. This classification system may be the most natural for a composer, who is primarily interested in the aesthetics of a particular sound transformation. Table 1 shows a selection of effects and the modified perceptual attributes [14][4][7]. For each effect main attributes are identified and additionally one or more other attributes that are modified by to a lesser extent. The perceptual attributes comprise of *loudness*, *duration* and *rhythm*, *pitch* and *harmony*, *timbre* and *quality*, and *space*. Listing 2 shows the echo effect class definition in the Audio Effects Ontology. The ontology contains a class `fx:Fx` representing the superclass for effect classification. Subclasses, such as `fx:SpatialFx` and `fx:LoudnessFx`, are linked to the main perceptual attributes with a restriction on the predicate `fx:main_attribute`. Subclasses of these classes in turn inherit these attributes, hence we only directly link the effect to secondary perceptual attributes using restrictions on the predicate `fx:other_attribute`. The perceptual attributes are defined as individuals of the class `fx:PerceptualAttribute`.

DAFx Name	Perceptual Attribute	
	Main	Other
Distance Change	S	L,T
Directivity	S	P,T
Echo	S	L
Granular Delay	S	L,D,P,T
Panning	S	
Reverberation	S	L,D,T
Rotary Speaker	S	P,T
Filter	T	L
Comb Filter	T	L,P
Equaliser	T	L
Ring Modulation	P,T	
Robotisation	P,T	L
Spectral Tremolo	L,T	D
Spectral Warping	T,P	L
Time Shuffling	L,D,P,T	
Vibrato	L,P	T,D

Table 1: Selection of digital audio effects and affected perceptual attributes (L: loudness, D: duration and rhythm, P: pitch and harmony, T: timbre and quality, S: space)[7][14].

A technical classification based on implementation techniques on the other hand is not as straight-forward. However, such a taxonomy would be helpful for the developer interested in the relationships of effects based on underlying digital signal processing (DSP) algorithms. Verfaillie et al. [7] proposed a technical classification based on [15]:

⁵<http://dublincore.org/documents/dcmi-terms/>

⁶OWL Reference: <http://www.w3.org/TR/owl-ref/>

⁷<http://purl.org/NET/c4dm/event.owl/>

⁸<http://purl.org/NET/c4dm/timeline.owl/>

⁹The Multitrack Ontology <http://purl.org/ontology/studio/multitrack>

- filters
- delays (resampling)
- modulators and demodulators
- nonlinear processing
- spatial effects
- time-segment processing
- time-frequency processing
- source-filter processing
- spectral processing
- time and frequency warping

```

fx:SpatialFx a owl:Class ;
    rdfs:subClassOf fx:Fx ,
    [ rdf:type owl:Restriction ;
      owl:onProperty fx:main_attribute ;
      owl:hasValue fx:Spatial
    ] .

fx:Echo a owl:Class ;
    rdfs:subClassOf fx:SpatialFx ,
    [ rdf:type owl:Restriction ;
      owl:onProperty fx:other_attribute ;
      owl:hasValue fx:Loudness
    ] .

```

Listing 2: Description of the echo effect in the Audio Effects Ontology (perceptual).

Naturally, this type of classification has limits; some audio effects, e.g. pitch shifting, can be implemented by different techniques, thus some ambiguity is inherent in this system. Furthermore, one can argue that *spatial effect* is not an implementation technique as such, as it relates primarily to a modified perceptual attribute. In order to provide a detailed ontology for the technical description of audio effects a DSP ontology is desirable, and constitutes future work in this field of research.

In addition to the classification systems described above, we propose the development of a taxonomy from an audio engineering point of view. Here, it is important to clearly define the meaning of the term "audio effect". While in our research we mostly use the term in its general sense, equating the terms *audio effect* and *sound transformation*, from an audio engineer's point of view often a distinction is made between *effects* and *processors*. Such a classification system could be seen as lying in between a perceptual and technical system, based on the audio effects' roles in the production workflow. In audio engineering *effects* are more artistic in nature, altering the sound in a dramatic way, for instance a tapped delay or a flanger. Audio *processors* on the other hand are transformations aimed at the enhancement of sound mostly in post-production and mastering. Equalisers and certain compressors fall into this category. The online database for DAFx plugins by KVR Audio¹⁰ lists *Mastering* as a separate category, containing mostly non-linear effects, such as enhancers and compressors designed to be applied on a musical mixture. A future direction of this research is a system unifying the audio effect ontologies of different disciplines, thus enabling and improving communication between developers, composers and audio engineers, providing a basis for the development of software agents processing audio effects related information to assist interdisciplinary work. Although we want to

describe digital audio effects, we consider an effect as such as an acoustical phenomenon (e.g. an *Echo* is a series of reflections of a sound) which can then be linked to an implementation with its respective algorithm as its physical manifestation in order to describe audio effects software or transformations. This approach also distinguishes the ontology from the LV2 (Linux Audio Developers Simple Plugin API version 2) specification¹¹, which, although also written in RDF and containing a classification scheme, is limited to the description of effects implemented in LV2, without discerning implementations and audio effects as physical phenomena.

4.2. Effect Transformations

The Audio Effects Ontology also defines concepts for describing the application of effects to a signal. The class `fx:Transform` is used here, comparable to the way the Vamp Ontology¹² defines concepts for the application of feature extraction plugins. An `fx:Transform` may be linked to an effect implementation using `fx:Implementation`. This implementation class may also act as the connection to the taxonomy by linking it to `fx:Fx` with the property `fx:implementation_of` (an inverse property linking an effect type to an implementation is also given by `fx:implementation`). Using this system we are able to associate events on the audio signal timeline as defined by the Music Ontology to a particular transform, which in turn is associated to information about the implementation, the effect type, and the modified perceptual attributes. Moreover, we may also describe an adaptive effect implementation that processes metadata in the form of RDF data, e.g. note onsets described with concepts from the Audio Features Ontology¹³[7][6]. We may describe the parameters of a sound effect implementation, both, to describe the settings at a particular transform, and to describe the available parameters for a given effect implementation. A set of standard parameters, such as the *dry/wet mix*, *feedback amount* and *gain* are defined in the ontology as well, which may be linked from an implementation parameter. In summary, the FXO is capable of describing audio effects in taxonomical systems adapted to different disciplines, effect implementations (e.g. plugins and hardware devices), and the application of audio transformations to audio signals.

5. QUERYING METADATA

Semantic metadata in RDF makes it possible to perform complex queries over the data using a query language such as SPARQL¹⁴, assuming the metadata is accurately accumulated during the production process. Listing 3 shows RDF data describing an onset event on a signal timeline created by the application of an echo effect, in this case "RDFx_Delay_1". The description includes the parameter settings of the effect, as well as provenance data about the origin of the audio event in the multitrack project during production. Running the SPARQL query from listing 4 over the metadata associated with the resulting mixture returns all events created by audio effects modulating loudness as "other" perceptual attribute as shown in table 1. We may also query for additional information, such as plugin name or developer. The example shows that with this system we can retrieve information from a musical mixture otherwise lost during the mixing process, or only

¹¹<http://lv2plug.in/ns/lv2core>

¹²<http://www.omras2.org/VampOntology/>

¹³http://motools.sourceforge.net/doc/audio_features.html/

¹⁴<http://www.w3.org/TR/rdf-sparql-query/>

¹⁰<http://www.kvraudio.com/get.php>

```

:transform_0 a fx:Transform;
fx:parameter_set [ fx:identifier "dryWetMix" ;
  fx:value "50"^^xsd:float ] ,
[ fx:identifier "delayTime" ;
  fx:value "0.297"^^xsd:float ] ,
[ fx:identifier "feedback";
  fx:value "0"^^xsd:float ] ;
fx:transform :rdfx_delay.

:event_0 a af:Onset ;
event:time [ a tl:Instant ;
  tl:at "PT1.897007S"^^xsd:duration ;
  tl:onTimeLine :signal_timeline_0 ] ;
fx:created_by_fx :transform_0 ;
fx:track_origin :GuitarTrack .

:example_multitrack a mt:MultitrackProject ;
mt:track :DrumTrack, :GuitarTrack .

```

Listing 3: RDF data describing an onset event on a signal timeline created by the application of the effect "RDFx_Delay_1".

retrievable by means of feature extraction from source-separated signals. However, tracking metadata throughout the production process makes source separation redundant in this context. Moreover, feature extraction from noisy audio material may pose problems concerning accuracy (this has been shown for the case of reverbation) [16].

```

SELECT ?time ?track WHERE {
  ?a event:time ?b ;
    fx:created_by_fx ?c ;
    fx:track_origin ?track.
  ?b tl:at ?time .
  ?c fx:transform ?d .
  ?d fx:implementation_of ?e .
  ?e fx:other_attribute fx:Loudness .
}

```

Listing 4: SPARQL query retrieving all events created by a transformation modulating loudness as the secondary perceptual attribute. *Track* represents the original audio track in the multitrack production prior to the mixdown.

By integrating a query engine in an audio production system processing metadata according to the proposed ontology, the user is not only capable to select audio effects by semantic descriptors of different domains, but is also able to retrieve detailed metadata about workflows and techniques concerning audio productions.

6. CONCLUSIONS

In this paper we presented an ontology defining concepts for the application and classification of digital audio effects. We showed how this novel ontology fits in to existing Semantic Web ontologies, particularly the Music Ontology and the Studio Ontology extension. We showed that RDF metadata accumulated during the production process using the Studio Ontology Framework allows for the retrieval of detailed information about a music piece. The information may be used for musicological purposes revealing production workflows and the origin of individual audio events, or for further processing by passing it to content-based (adaptive) audio effects or feature extractors.

Our investigation of audio effects with respect to classification and ontology design showed that it is necessary to create multiple ontologies covering the different disciplines concerned (e.g. classifications based on implementation for developers or on perceptual attributes for composers). The presented work reveals the need for more specialised ontologies for our music information management framework, such as a dedicated signal processing ontology. Future work includes the development of such ontologies and further development and integration of the developed tools in music production applications.

7. REFERENCES

- [1] E. Brazil, "Cue point processing: An introduction," *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01)*, Limerick, Ireland, December 6-8, 2001.
- [2] E. Gomez, G. Peterschmitt, X. Amatriain, and P. Herrera, "Content-based melodic transformations of audio material for a music processing application," in *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFX-03)*, London, UK, September 8-11, 2003.
- [3] E. Pampalk, P. Hlavac, and P. Herrera, "Hierarchical organization and visualization of drum sample libraries," in *Proc. of the 7th Int. Conference on Digital Audio Effects (DAFX-04)*, Naples, Italy, October 5-8, 2004.
- [4] V. Verfaillie, U. Zölzer, and D. Arfib, "Adaptive digital audio effects (A-DAFx): A new class of sound transformations," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 5, 2006.
- [5] G. Fazekas and M. Sandler, "Intelligent editing of studio recordings with the help of automatic music structure extraction," *presented at the AES 122nd Convention*, Vienna, Austria, 2007.
- [6] T. Wilmering and M. Sandler, "RDFx: Audio effects utilising musical metadata," *presented at the 4th IEEE International Conference on Semantic Computing*, Pittsburgh, PA, USA, 2010.
- [7] V. Verfaillie, C. Guastavino, and C. Traube, "An interdisciplinary approach to audio effect classification," *Proceedings of the 9th International Conference on Digital Audio Effects (DAFX-06)*, Montreal, Canada, September 2006.
- [8] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, May 2001.
- [9] I. Horrocks, "Ontologies and the Semantic Web," *Communications of the ACM*, vol. 51, no. 12, pp. 58–67, 2008.
- [10] T. R. Gruber, "A translation approach to portable ontology specification," *Knowledge Acquisition*, vol. 5, no. 2, 1993.
- [11] G. Fazekas, Y. Raimond, K. Jakobson, and M. Sandler, "An overview of Semantic Web activities in the OMRAS2 Project," *Journal of New Music Research special issue on Music Informatics and the OMRAS2 Project*, vol. 39 issue 4, pp. 295–311, 2010.
- [12] Y. Raimond, S. A. Abdallah, M. Sandler, and F. Giasson, "The music ontology," *Proceedings of the International Conference on Music Information Retrieval*, 2007.
- [13] G. Fazekas, Y. Raimond, and M. Sandler, "A framework for producing rich musical metadata in creative music production," *presented at the AES 125th Convention*, San Francisco, CA, USA, 2008.
- [14] X. Amatriain, J. Bonada, À. Loscos, J. L. Arcos, and V. Verfaillie, "Content-based transformations," *Journal of New Music Research*, vol. 32, no. 1, pp. 95–114, 2003.
- [15] U. Zölzer, *DAFX - Digital Audio Effects*, J. Wiley & Sons, 2002.
- [16] T. Wilmering, G. Fazekas, and M. Sandler, "The effects of reverberation on onset detection tasks," *presented at the AES 128th Convention*, London, UK, 2010.

IDENTIFICATION OF TIME-FREQUENCY MAPS FOR SOUNDS TIMBRE DISCRIMINATION

Anaïk Olivero

Laboratoire de Mécanique et d'Acoustique,
UPR 7051, CNRS
Marseille, France
olivero@lma.cnrs-mrs.fr

ABSTRACT

Gabor Multipliers are signals operator which are diagonal in a time-frequency representation of signals and can be viewed as time-frequency transfer function. If we estimate a Gabor mask between a note played by two instruments, then we have a time-frequency representation of the difference of timbre between these two notes. By averaging the energy contained in the Gabor mask, we obtain a measure of this difference. In this context, our goal is to automatically localize the time-frequency regions responsible for such a timbre dissimilarity. This problem is addressed as a feature selection problem over the time-frequency coefficients of a labelled data set of sounds.

1. INTRODUCTION

Given a pair of sound signals, our approach yields an estimate of a time-frequency transfer function to go from one sound to another. Our goal, in the present paper, is to further analyze this time-frequency transfer function. In particular we want to identify the regions in the time-frequency domain which carry most discriminant information, in the context of sounds categorization.

The approach proposed in [1] for the analysis and categorization of families of sound signals, exploits the transformation between signals in the family. In this method, the signals are supposed to be similar enough in the time-frequency domain so that these transformations can be modeled as Gabor multipliers, i.e. linear diagonal operator in a Gabor representation (subsampled version of Short Time Fourier Transform). Gabor multipliers are characterized by a *time-frequency transfer function*, hereafter called *Gabor mask*. Gabor Multipliers estimation have been studied in [2], [3], in the context of sounds transformation. In [1], Gabor masks were used to categorize sounds, by means of a corresponding complexity measure, on the basis of pairwise comparisons. Such estimated transfer functions can be viewed as a vector of features characterizing the differences between two signals. We have shown in [1] that a well chosen average the values of these features could yield sensible classifications within controlled musical signal families.

The timbre [4] is a relative notion defined as the difference between two sounds with same pitch, duration and loudness. We aim to automatically identify the time-frequency regions which have been responsible for a given timbre difference and propose a method for this task. In the context of harmonic sounds of musical instruments, it is well known [5] that the timbre can be characterized by time and spectral descriptors (such as attack time, spectral centroid, spectral flow,...). These sounds descriptors are implicitly

captured in the time-frequency representation of a signal and so their differences are carried by the Gabor masks.

In the context of sounds synthesis, such a time-frequency significance map can be useful, as it gives up the time-frequency representation regions of interest for synthesizing a sound into another, as the authors in [6] who explain the importance of the control of the signals descriptors in the context of sounds morphing.

In Section 2, we present the general setting and describe the basic concepts of signal representation and time-frequency analysis we shall be working with. The feature selection problem is investigated in Section 3. Some examples of transfer functions between notes and the estimated time-frequency regions are given in Section 4 from three different instrument families.

2. GABOR FRAMES AND GABOR MULTIPLIERS

In the finite-dimensional situation \mathbb{C}^L , the Short Time Fourier Transform of the signal can be seen as the analysis map of a Gabor frame representation of the signal, as explained in [7]. A Gabor frame is an overcomplete family of time-frequency atoms generated by translation and modulation on a discrete lattice of a mother window, denoted by $g \in \mathbb{C}^L$. These atoms can be written as

$$\pi_{mn}g[l] = g_{mn}[l] = e^{2i\pi mb(l-na)}g[l-na], \quad (1)$$

where a and b are two positive integers, such that L is multiple of both a and b and (a, b) generates a time-frequency lattice. π_{mn} is a time-frequency shift operator. Here, all operations have to be understood modulo L . We set $M = L/b$ and $N = L/a$.

The time-frequency representations of signal x is given by

$$X[m, n] = \langle x, g_{mn} \rangle$$

In particular there are situations (called tight) where the inversion takes a particularly simple form, the analysis and synthesis windows are the same and the reconstruction is given by $x = \sum_{m,n} X[m, n]g_{mn}$. Gabor transforms give a frame framework to the time-frequency representations. In this context, a signal transformation can be constructed by pointwise multiplication between the analysis coefficients and a *transfer function*, followed by the reconstruction with the synthesis window. Such transformations are generically called *multipliers*. Denoting by \mathbf{m} the transfer function, we shall denote by $\mathbb{M}_{\mathbf{m}}$ the corresponding multiplier such that

$$\mathbb{M}_{\mathbf{m}}x = \sum_{m,n} \mathbf{m}[m, n]X[m, n]g_{mn}. \quad (2)$$

Let x_i and x_j denote the input and output signals, respectively. We assume the following model

$$x_j = \mathbb{M}_{\mathbf{m}} x_i + \epsilon,$$

where ϵ represent perturbations, modeled as additive gaussian noises, and \mathbf{m} is an unknown Gabor mask, which we want to estimate. A possible solution is obviously $\mathbf{m} = X_j/X_i$, where X denote the Gabor transform of x , but such a solution is not bounded in general. We prefer to turn to a regularized least squares solution. More precisely, we seek $\mathbf{m} \in \mathbb{C}^{M \times N}$ which minimizes the expression

$$\Phi[\mathbf{m}] = \|x_j - \mathbb{M}_{\mathbf{m}} x_i\|^2 + \mu r[\mathbf{m}], \quad (3)$$

where $r[\mathbf{m}]$ is a regularization term, whose influence on the solution is controlled by the parameter μ .

The formulation (3) involves a non diagonal matrix, where the non diagonal terms arise from the correlations between the atoms of the representation. A first approach is to formulate the problem directly in the transform domain, or equivalently to a reduction of the problem (3) to its diagonal.

$$\tilde{\Phi}[\mathbf{m}] = \|X_j - X_i \mathbf{m}\|^2 + \mu r(\mathbf{m}), \quad (4)$$

Such an approximation has the advantage to admit a closed form expression for its unique minimizer. For example, we can choose $r(\mathbf{m}) = \|\mathbf{m} - \mathbf{m}^t\|_F^2$, where \mathbf{m}^t is a given target time-frequency function that will help to design the estimated Gabor mask \mathbf{m} . The time-frequency function \mathbf{m}^t can be useful in the context of sound morphing, where we aim to “interpolate” between two sound signals. Given \mathbf{m}^t , we obviously obtain a regularized solution for \mathbf{m} which reads

$$\tilde{\mathbf{m}} = \frac{\overline{X_i} X_j + \mu \mathbf{m}^t}{|X_i|^2 + \mu},$$

3. TIME-FREQUENCY CHARACTERIZATION OF A SOUNDS CATEGORIZATION

3.1. A divergence between two spectra

The Itakura Saito divergence is often used to compare two audio spectra in the context of speech processing [8]. This measure is expressed as

$$d_{IS}(|X_j|, |X_i|) = \sum_l \frac{|X_j[l]|}{|X_i[l]|} - \log \frac{|X_j[l]|}{|X_i[l]|} - 1$$

where $|X_i|$ and $|X_j|$ are the magnitude of signals spectrum or signals time-frequency spectrum and l denotes a frequency or a time-frequency bin. The Itakura Saito divergence is not symmetric and a symmetrized version [9] can be derived as

$$d_{SIS}(|X_j|, |X_i|) = \frac{d_{IS}(|X_j|, |X_i|) + d_{IS}(|X_i|, |X_j|)}{2} \quad (5)$$

We first denote that Equation (5) is not bounded in general and a way to avoid such a problem is to regularize it. If we denoted by \mathbf{m}_{ij} the Gabor mask obtained by a diagonal approximation regularized with $r(\mathbf{m}) = \|\mathbf{m} - 1\|_2^2$ between signals x_i and x_j , then $d_{SIS}(|\mathbf{m}|, 1)$ is a natural choice two compare two spectra as the masks are more stable than the quotient of two spectra.

The choice of the regularization term r was motivated by the desire of maintain $\mathbf{m} = 1$ as reference, corresponding to “no transformation”. However, given that Gabor transforms of real valued

signals are complex valued, and that the phase of the Gabor transform is generally difficult to handle precisely, the reference choice may be $|\mathbf{m}| = 1$ rather than $\mathbf{m} = 1$. This suggests the use of a regularization term of the form $r(\mathbf{m}) = \|\mathbf{m} - 1\|^2$. This leads to an explicit expression for the Gabor mask given by

$$|\mathbf{m}_{ij}| = \frac{|X_i X_j| + \mu}{|X_i|^2 + \mu}.$$

Then, the phase of the Gabor mask is given by the phase difference between X_j and X_i .

3.2. A time-frequency map of the information responsible for the categorization

First, the Itakura Saito divergence is separable and if we define

$$d_{ij}[m, n] = \frac{1}{2} (|\mathbf{m}_{ij}[m, n]| - \log |\mathbf{m}_{ij}[m, n]| - 1 + |\mathbf{m}_{ji}[m, n]| - \log |\mathbf{m}_{ji}[m, n]| - 1) \quad (6)$$

then the symmetrized Itakura Saito divergence reads

$$d_{SIS}(|\mathbf{m}|, 1) = \frac{1}{MN} \sum_{m,n} d_{ij}[m, n] \quad (8)$$

The dissimilarity matrix $d[m, n]$ represents the ability of a time-frequency bin to discriminate two given classes and gives us a dissimilarity measure between two sounds for each time-frequency bin. We see in Equation (8) that the information carried by $d_{SIS}(|\mathbf{m}|, 1)$ is drastically reduced, as we just consider the sum over all the time-frequency coefficients. We also propose to use a weighted Itakura Saito divergence as

$$d_{SIS}^\alpha(|\mathbf{m}|, 1) = \sum_{m,n} \alpha_{mn} d[m, n] \quad (9)$$

where the α are the weights, which indicate the relevance of each time-frequency bin and are to be estimated from data. These weights are supposed to emphasize one subset of time-frequency bins over the others. Then, we impose the following properties : $\alpha_{mn} \geq 0$ and $\sum_{mn} \alpha_{mn} = 1$, so that the Equation (8) can be viewed as a uniform version of the Equation (9).

We propose to model our problem in the spirit of the Relief algorithm [10], a feature weighting algorithm that iteratively selects feature over a training data set. We suppose that we have a training data set of labelled signals $\{x_i, i = 1..N\}$ composed by two different classes of signals, where the first class contains N_1 signals, and the second contains N_2 signals. We denote by \mathcal{C}_i the set of indices of the signals which are in the class of the signal x_i . We want to define a distance that discriminates the 2 classes as clearly as possible. We can formally model this problem as the maximization of a margin, where the margin is given by:

$$\rho(\alpha) = \sum_{mn} \alpha_{mn} \left(\sum_i \sum_{j \notin \mathcal{C}_i} d_{ij}[mn] - \sum_i \sum_{j \in \mathcal{C}_i} d_{ij}[mn] \right)$$

In other words, the margin is considered as measure of the ability of a set of weights to discriminate two classes of signals.

For the sake of clarity, let us define

$$\begin{aligned} z_{mn} &= \left(\sum_i \sum_{j \notin \mathcal{C}_i} d_{ij}[mn] - \sum_i \sum_{j \in \mathcal{C}_i} d_{ij}[mn] \right) \\ &= (\langle D^-, d[m, n] \rangle - \langle D^+, d[m, n] \rangle) \end{aligned}$$

The matrices D^+ and D^- are in $\{0, 1\}^{N \times N}$ and are used to represent the repartition of the data in two different classes. $D_{ij}^+ = 1$ if i and j are in the same class and 0 otherwise, whereas $D_{ij}^- = 0$ if i and j are in the same class and 1 otherwise. The maximization of the margin emphasize the data which are in the same class. Now, the problem takes the form of a minimization under constraints

$$\max_{\alpha} \alpha^T z \text{ s.t. } \|\alpha\|_2^2 = 1 \text{ and } \alpha \geq 0 \quad (10)$$

The solution to this problem is given by : $\alpha = \frac{z^+}{\|z^+\|_2}$, where $z^+ = [\max(z_{mn}, 0)]_{mn}$.

This problem implicitly contains sparsity constraints as it reduces the time-frequency information, by removing the negative values of z . The removed time-frequency bins can also be viewed as irrelevant for the given classification task. The α values also give us the importance of a given time-frequency coefficient in our given task. Other algorithms performs such a feature selection and we refer to [11] for a review.

4. EXPERIMENTS

The time-frequency maps given by the coefficients $\{\alpha_{mn} : m, n\}$ allows to identify the time-frequency information responsible for a classification task. They provide an average map of the differences between each class, with less variability compared to the gabor masks obtained between individual pairs of sounds. This is also a generic way to automatically enlighten the time-frequency differences of timbre between two classes of harmonic sounds, as we suppose no signal model and no descriptors choices. Here, we argue that these time-frequency maps generalize the information contained in the Gabor Masks by pairwise comparison of two classes of sounds, which can be more useful in the context of sounds morphing, as they can be used to transform a sound from one class to another. As we will see below, the time-frequency differences will depend on the sounds classes we are comparing.

Some experiments are shown here. We used three classes of musical instrument sounds playing the same note, with fundamental frequency $f_0 = 196$ Hz (G3) : 16 clarinets, 15 saxophones (8 alto and 10 tenors) and 13 trumpets. Prior to mask estimation, the signals are adjusted so that their onset coincide, as the onset time is not relevant in our task. Now, all the sounds are supposed to have a good time-frequency alignment, so that the Gabor mask capture a pertinent information. In each experiment, a data set contains the sounds of two different classes. We considered three different data sets : the clarinets and the saxophones, the clarinets and the trumpets, the trumpets and the saxophones. The spectrograms of one sound of each class are shown in Figure 1, obtained using a Hanning mother window and parameter values $M = 512$, $a = 64$ and displayed in a logarithmic amplitude scale.

The time-frequency maps for the three data sets are computed as explained in Section 3.2 and shown in Figure 2. As expected, we can see that the three instruments classes present some time-frequency differences at different locations and these differences can be interpreted physically. All time-frequency maps exhibit a harmonic structure supply by a formantic structure, which is coherent with our understanding of the acoustic of these musical instruments. Each map emphasize the differences between two classes of sounds. For example, the even harmonics (which are known to be a relevant clue for identifying the clarinets) appear strongly in the clarinets/saxophones and clarinets/trumpets maps. However, their importance in the classification process differs slightly when

the clarinets are compared to trumpets or saxophones. The maps also reveal how the frequency content during the attack differs according to the two classes we are observing. This information can be particularly useful in practice to distinguish the trumpets from the clarinets and saxophones classes.

5. CONCLUSIONS AND PERSPECTIVES

We have described in this paper a method for better exploiting the information contained in time-frequency masks estimated from families of sound. Namely, the proposed approach is able to retrieve the sub-domains in the time-frequency plane that permit discrimination of two instrument sounds playing the same note, in other words the time-frequency information carrying the timbre differences. This goal is achieved by coupling mask estimation with using a feature selection method on a labelled class of sounds.

Further developments of this work will involve the construction of smoother versions of the time-frequency map, and applications in a context of sounds morphing.

6. ACKNOWLEDGMENTS

Many thanks to Richard Kronland-Martinet and Bruno Torr sani for their help in this work.

7. REFERENCES

- [1] Ana k Olivero, Laurent Daudet, Richard Kronland-Martinet, and Bruno Torr sani, "Analyse et cat gorisation de sons par multiplicateurs temps-fr quence," in *XXIIe colloque GRETSI (Dijon)*, 8-11 septembre 2009.
- [2] P. Depalle, R. Kronland-Martinet, and B. Torr sani, "Time-frequency mutlipliers for sound synthesis," in *Proceedings of the Wavelet XII conference, SPIE annual Symposium*, San Diego, 4-8 September 2007, pp. 221–224.
- [3] Ana k Olivero, Bruno Torr sani, and Richard Kronland-Martinet, "A new method for gabor multipliers estimation : Application to sound morphing," in *EUSIPCO 2010*, Alborg, Denmark, August 2010, pp. 507–511.
- [4] Acoustical Terminology, ASA, *American Standards Association*, New York, 1960.
- [5] G. Peeters, S. McAdams, and P. Herrera, "Instrument sound description in the context of mpeg 7," in *Proc. ICMC*, Berlin, Germany, August. 27- Sept. 1 2000, pp. 203–206.
- [6] Marcelo Caetano and Xavier Rodet, "Automatic timbral morphing of musical instruments sounds by high-level descriptors," in *Proc. ICMC 2010*, pp. 11–21.
- [7] Peter S ndergaard, *Finite Discrete Gabor Analysis*, Ph.D. thesis, Vienna University, 2007.
- [8] R. Gray, A. Buzo, Jr. Gray, A., and Y. Matsuyama, "Distortion measures for speech processing," aug 1980, vol. 28, pp. 367 – 376.
- [9] B. Wei and J. D. Gibson, "Comparison of distance measures in discrete spectral modeling," in *Proc. 9th DSP Workshop 1st Signal Processing Education Workshop*, Oct. 15-18, 2000.

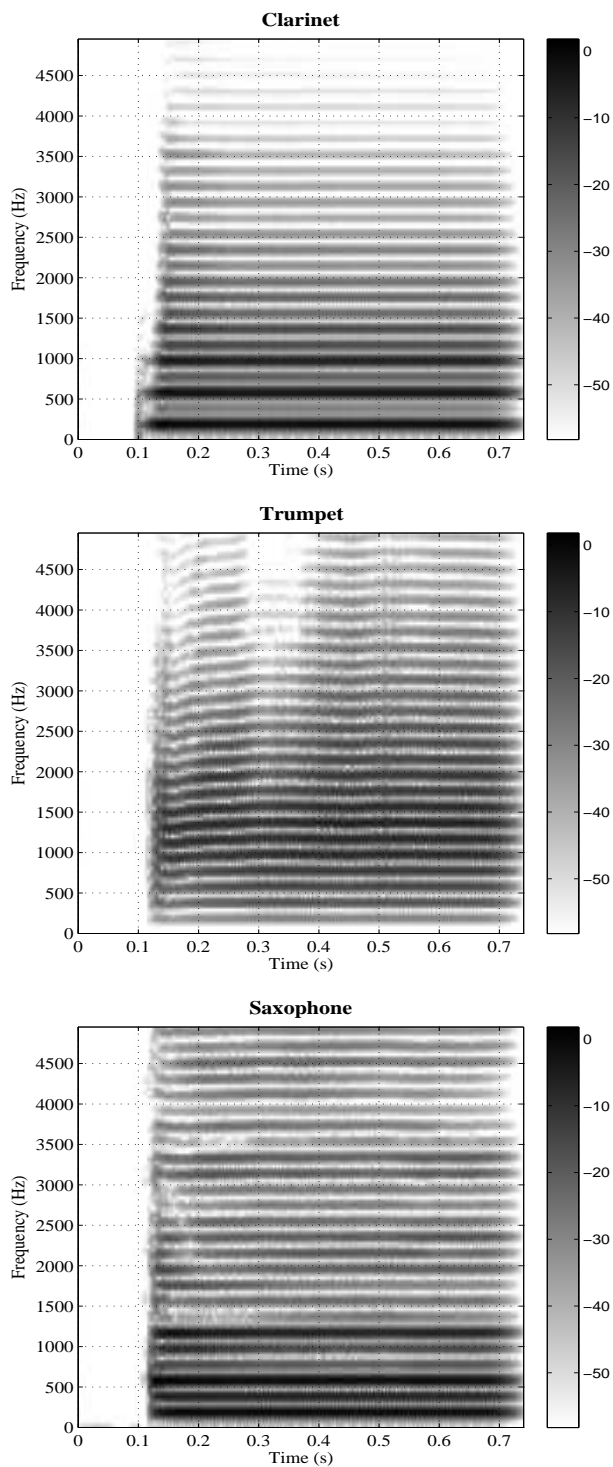


Figure 1: Three spectrograms of our sounds data set

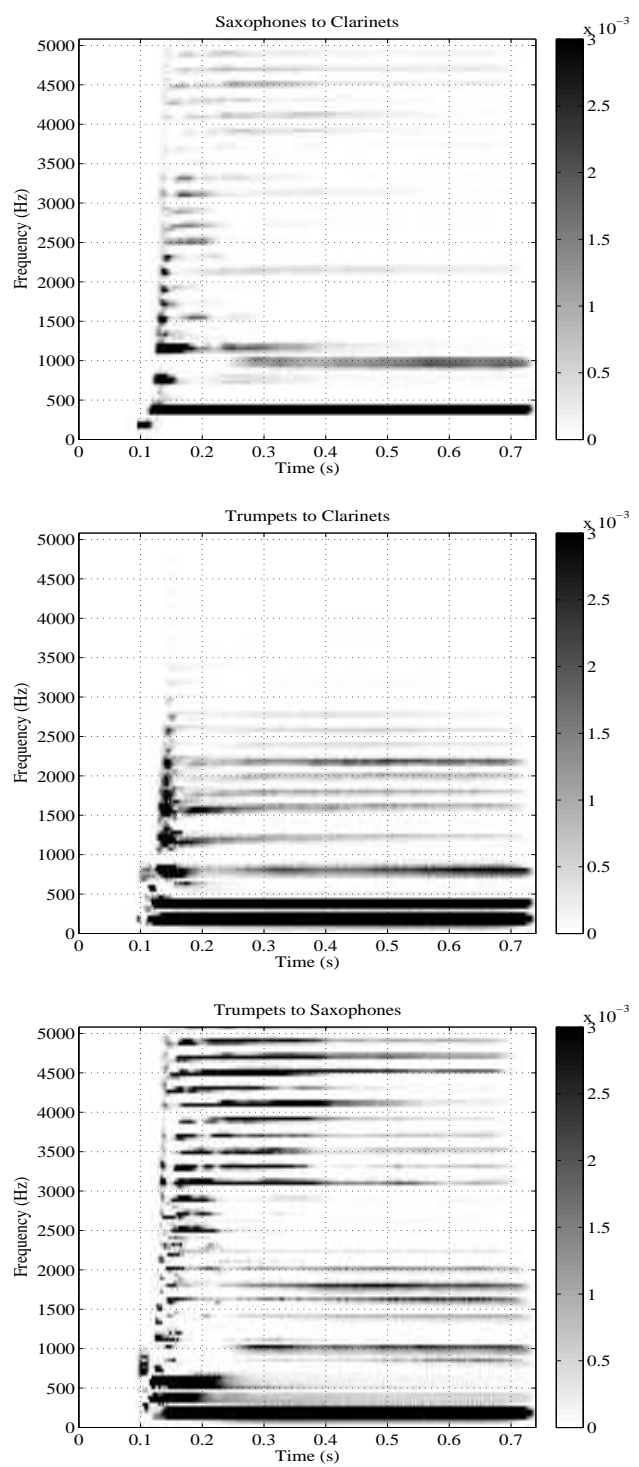


Figure 2: The three time-frequency maps α obtained from our data set by pairwise comparison of three classes

- [10] Yijun Sun, "Iterative relief for feature weighting: Algorithms, theories, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, No 6, pp. 1035–1051, June 2007.

- [11] A. Bagherjeiran and C.F. Eick, *Studies in Computational Intelligence (SCI) 73*, chapter Distance Function Learning for Supervised Similarity Assessment, Springer-Verlag, 2008.

COMBINING CLASSIFICATIONS BASED ON LOCAL AND GLOBAL FEATURES: APPLICATION TO SINGER IDENTIFICATION

Lise Regnier,

IRCAM
Paris, France

`lise.regnier@ircam.fr`

Geoffroy Peeters,

IRCAM
Paris, France

`geoffroy.peeters@ircam.fr`

ABSTRACT

In this paper we investigate the problem of singer identification on acapella recordings of isolated notes. Most of studies on singer identification describe the content of signals of singing voice with features related to the timbre (such as MFCC or LPC). These features aim to describe the behavior of frequencies at a given instant of time (local features). In this paper, we propose to describe sung tone with the temporal variations of the fundamental frequency (and its harmonics) of the note. The periodic and continuous variations of the frequency trajectories are analyzed on the whole note and the features obtained reflect expressive and intonative elements of singing such as vibrato, tremolo and portamento. The experiments, conducted on two distinct data-sets (lyric and pop-rock singers), prove that the new set of features capture a part of the singer identity. However, these features are less accurate than timbre-based features. We propose to increase the recognition rate of singer identification by combining information conveyed by local and global description of notes. The proposed method, that shows good results, can be adapted for classification problem involving a large number of classes, or to combine classifications with different levels of performance.

1. INTRODUCTION

The goal of classification is to assign unlabeled patterns into a number of known categories. A system of classification is based on an appropriate description of the patterns (features) and on a statistical algorithm (classifier) trained to learn the specificities of each pattern for the given problem. To evaluate the performance of a system, a new set of data is given to the classifier and the portion of patterns assigned to their correct class is given as an indicator of its global performance. However, each system of classification has its own limitation. To increase the classification accuracy it is necessary to introduce and combine complementary information on the problem (either new representations of the patterns or new specifications of the classes).

Classification of speech and musical signals has been largely investigated this last decade and all sort of features (temporal and spectral) and classifiers have been tested. A special attention has been given to features related to the timbre. Timbre is a perceptual attribute of the sounds that seems to be multi-dimensional. However, research has shown that timbre can be partly transcribed by the spectral envelope of sounds. In speech processing area, the source-filter model [1] clearly justify the use of spectral envelope for speech recognition problem. Researches on instruments recognition have also proved that spectral envelope was a good element to discriminate musical instruments. The singing voice is a musical instrument that has much in common with speech. For this

reason, most of works carried out on the topic of singer identification have based the description of sung signals on features derived from the spectral envelope. They have obtained satisfying results with this approach but to improve the identification performance it is necessary to extract additional information on the signals of singing voice.

In this study we suggest to describe signals of singing voice with intonative and expressive elements characteristic of singing. We propose a new set of features derived from the analysis of the trajectory of the fundamental frequency (and its harmonics). In a previous work [2] we have demonstrate that elements such as vibrato, tremolo and portamento were efficient to detect the presence of singing voice within a song. We propose now to evaluate if these features can be used to discriminate singers between them. More precisely, we evaluate if intonative features can be combined with timbre-based features to improve the performance of singer identification.

The combination of information is not a straightforward problem. In our case, the patterns to be classified are notes sung acapella. For each note, we extract timbre-based and intonative features. Timbre-based features are computed on short frames (local features) whereas information on intonation is obtained when considering the note globally (global features). As a result, the two descriptions have different sizes and cannot be compacted into a single feature-set without adding redundancy or deleting important information. The only solution is then to train two classifiers on each set of features separately and to combine their decisions afterwards. Working with only two decisions, simple voting methods cannot be applied. In addition we know from preliminary experiments, that timbre-based features provide much better results than intonative features. In general it is ticklish to improve a good performance by combining information less accurate. The proposed combination method is based on the class set reduction approach. It starts with the feature set leading to the best performance. The output of the classifier for this feature set is analyzed to deduce a restricted set of possible classes. The deduction is done by regarding the membership value (pseudo-posterior probability) for all the classes. The second set of feature is then used to perform the classification within the reduced set of classes. The membership values, for the remaining classes, returned by the two classifiers are then analyzed to take the final decision.

The paper is organized as follow: In section 2, we present some related works on singer identification and on combination of information. We present in section 3 the different elements of our method: the features, the classifiers and the combination method. The method is evaluated in section 4 on two data-sets composed of accapella recording of notes. Finally, section 5 summarizes the main results of the paper and offer some conducting remarks.

2. RELATED WORKS

2.1. On singer identification

Numerous researches have been carried out on the topic of singer identification (SID) because the voice is for many listeners the element that focuses the most their attention. Most of the methods propose to describe and recognize the content of audio signals with spectral features such as MFCC ([3], [9], [6]), LPC and their variants ([4],[5]). The features are given as input to classifier to construct a model per singer present in the data-set. To retrieve the singer of a query song, the features extracted from this song are compared to the models obtained in the previous step and the song is assigned to the class whose model is the most likely. In previous researches SVM [3], GMM [4] [5] [6] [7] [8] ANN [3] [9] have been tested for the task of SID. We consider that models obtained using features extracted from audio mixtures (i.e. voice + instruments) represent the identity of the artist (or music band) instead of the singer. To get models more representative of the singer it has been suggested in [9] and [4] to perform the classification using the segments of the song where the voice is present only (i.e they discard the purely instrumental portions of the song from the analysis). These methods still rely on features extracted from audio mixtures and it is therefore not possible to examine how much the results obtained are corrupted by the presence of instrumental background. To obtain information directly related to the voice on mixtures it has been proposed in [7], [8] to deduce a solo-singer model from a model obtained on purely instrumental parts of the song and a model obtained on the vocals (voice+instruments) portions. In most of these studies, they do not find any improvement by treating separately instrumental and vocal parts of the song. We can suppose that in some cases, the performances of systems based on spectral features obtained on mixtures directly are strongly affected by the “album” or “produced effect” [10] (all songs from the same album/producer share overall spectral characteristics). It has also been suggested to perform the classification on isolated vocals: in [5] the voice is isolated by reducing the instrumental accompaniment, in [6] the voice is re-synthesized using the components harmonically related to the fundamental frequency of the sung melody. Some studies ([6] or [11]) have compared results obtained on acapella recordings with results obtained on voice isolated from mixtures (where the mixtures were created using the same acapella recordings mixed with other instrumental tracks). They usually reported that the performances obtained on isolated vocals is much lower than the performances obtained on acapella recordings and suggest that the loss is due to the artifacts created when isolating the voice.

In this study we work in the ideal case of acapella recordings and suggest performing identification by combining local and global descriptions of the voice. So, before presenting the details of our method (features and combination rules) we review in the next paragraph some of the basic points of the information combination theory for classification problem.

2.2. On combination of decisions

Each system of classification (features+classifier) has its own limitation. To improve classification accuracy it has been proposed to combine complementary information on the same problem. The best way to obtain complementary information on a problem is probably to describe the patterns with different approaches. In some ideal cases the feature-sets given by the different descrip-

tions can be directly combined to form a unique feature-set (early fusion). In many cases, when the features have different types, ranges of values, size or different physical meanings, grouping all the features together can completely degrade the information conveyed by the features when considered independently. For this reason, it has been proposed to combine the decisions of the systems of classification instead (late fusion). In this case, each classification system works with its own feature-set. Combination of decisions can be grouped into two categories according to their architecture: parallel and sequential.

In **parallel combination** all systems involved in the combination have to classify the same data-set into the same known categories. Then, the final decision is taken by applying predefined rules on the decisions of all the classifiers. A classifier can return: a *single class*, a *list of classes* ordered in term of preference or a *membership value* for each class [12]. From the membership values we can deduce the ranked list of classes, from the list we can deduce the most likely class. An overview of methods developed for each type of outputs is presented in [13]. The more complete outputs, the more difficult to combine. Theoretically, it is not feasible to combine membership measurements obtained using different feature spaces or different type of classifiers because their respective values may not have different significations as explained in [14]. However, the methods of transformation presented in [15] can be applied to normalize the outputs.

The goal of the combination is to reach a higher accuracy than each of the individual classifications. In practice, when all classifications have equivalent accuracies, most of the basic combination rules (as *majority voting* or *sum-rule*) can reach this goal as long as the classification are not too correlated. However, when the systems to be combined show different levels of performance it is necessary to introduce knowledge on the relative performances into the combination rule. An easy way to realize such a combination is to consider the classifiers outputs as a new features and to train the combination rule (or a meta-classifier). Methods based on trained combiners generally show good results, but to avoid a lack of generalization, a very large amount of training data is necessary. Indeed, if the combiner is trained on the data-set used to learn the specificities of classes there is a large risk of over-fitting. To avoid this, the data set should be divided into three parts. The models should be learned on the first part and evaluated with the second part of the data. The results obtained should then be used to train the combiner. The global performance should be evaluated on the remaining part.

In **sequential combination** the classification systems are applied one after another using the output of the previous classifier to define a new problem for the next classifier. The final decision is generally given by the last classification (the decision-making process can be viewed as a decision tree). From these sequential methods, we retain:

- The *hierarchical* methods [16] that can be applied when the data has a taxonomy,
 - the *cascade* methods [17] where a pattern is processed by a new classifier until it is classified with a certain degree of confidence
 - and the *multi-stage* methods [18] that attempt to reduce the number of possible classes at each stage until one class remain possible.
- Sequential methods are shown to be specially adapted to solve problem involving a large number of classes and are particularly suitable for the recognition of rare event (i.e when the classes of the data set are not well balanced). With sequential classification, there is no possible backwards analysis. If the decision taken at

one step is wrong the full process will be affected.

3. DETAILS OF THE PROPOSED METHOD

We first introduce our local and global features used to describe the sound samples, then we briefly present the classifiers used next in the evaluation. Finally, we present the combination rule developed to combine the different type of features.

3.1. Sound description

Any sound can be considered as a pattern varying along two dimensions: the time and the frequency axis (as shown by the common representation of sound: the spectrogram). To obtain an accurate description of a sound we suggest to describe sounds over these two dimensions: (1) describe behavior of the frequencies at a given time (on one frame of few ms) and (2) describe the temporal variations of one frequency (or one band of frequencies) on a interval of time (segment of few sec). Features extracted at a given time of the signal and repeated along the signal will be referred as *local features* and features extracted on a longer interval of time will be referred as *global features*. Local features have focussed the most attention in all audio classification problems. The relative amplitude of frequencies, represented by the overall shape of the spectrum (the spectral envelope), has been proved to be efficient to describe and discriminate sounds in tasks of speaker and musical instruments recognition. In this study we work on classification of sung signals. Singing voice differs from speech in his musical intention and also in its production. One of the major difference is that sung sounds are most of the time voiced and sustained to allow intelligibility of the lyrics. On these sustained voiced sound, many characteristics can be obtained by analyzing the temporal variations of one frequency band on a given interval of time. These variations, intentional or not, enhance the singing voice in two points: first, these variations add expression but also they help the voice to stand out of the instrumental background. In the next paragraph we present the features used to describe the spectral envelope on this study. Next, we present the features extracted on the frequency trajectories.

3.1.1. Timbre: Local description of sound

Information related to the timbre is supposed to be conveyed by the spectral envelope. This idea comes from the description of speech sound using the source filter model by Fant [1]. In this model we suppose that the source is a periodic train of pulses (where the pitch of the produced sound is given by the distance between the pulses) modified by a filter: the vocal tract. The goal is to decorrelate the filter from the source to obtain an approximation of the transfer function of the vocal tract. The vocal tract enhances some frequencies (phenomene of extra resonance). The response of the filter is given by the global shape of the spectrum: the spectral envelope.

Many methods, with different theoretical backgrounds, have been developed to estimate and encode the spectral envelope. In our evaluation, we use three different representations of the spectral envelope: the coefficients derived from the Linear Predictive Analysis (LPC), the Mel Frequency Cepstral Coefficients (MFCC) and the Cepstral Coefficients derived from the True Envelope (TECC).

LPC and MFCC have been already used for the task of singer recognition and for we refer the reader to the works presented in

2.1 for a detailed description of these coefficients. The true envelope, introduced in [19], has been mostly used in the speech processing area. As shown in [20], this envelope estimation is more robust (especially for high pitched signals) than many other envelope estimation methods. Like the MFCC, the true envelope is estimated in the cepstral domain. This domain offers the possibility to transform the convolution of two signals into the addition of their spectra. So that, the cepstrum of a speech signal is the addition of the cepstrum of the vocal tract response and the cepstrum of the excitation signal. The real cepstrum of a discrete signal $x(n)$ is defined as the inverse Fourier transform of the log-amplitude spectrum of $x(n)$. If $X(k)$ designates the k^{th} point of the discrete Fourier transform (DFT) of $x(n)$ (with K the total number of point of the DFT), the cepstrum $C(m)$ of $x(n)$ is given by:

$$C(m) = \sum_{k=0}^{K-1} \log(|X(k)|) e^{\frac{2i\pi km}{K}} \quad (1)$$

True envelope estimation is based on iterative cepstral smoothing of the log-amplitude spectrum. We denote $C_i(k)$ the cepstral representation of the envelope at the i^{th} iteration for the bin k of the DFT (1). The algorithm iteratively updates the smoothed input spectrum $A_i(k)$ using the maximum of the original spectrum $|X(k)|$ and the current spectral representation.

$$A_i(k) = \max(\log(|X(k)|), C_{i-1}(k)) \quad (2)$$

The cepstral smoothing is then applied to $A_i(k)$ to obtain $C_i(k)$. The iterative algorithm stops if for all bin k and a fixed threshold τ the relation $A_i(k) < C_i(k) + \tau$ is satisfied.

At the end of this operation, the true envelope has the same size than the cepstrum. To concentrate the information conveyed by this envelop into a smaller number of coefficients, the Discret Cosine Transform (DCT) is computed on the envelop and the firsts coefficients are retained. (This method is similar to the method applied to obtained the MFCC). We named in the following, the coefficients obtained TECC.

The goal of any envelope estimation is to retain from the signal the contribution of the filter by discarding information of the pitch. For high pitched signals this estimation can be problematic since the envelope can start following the peaks related to the pitch instead of the global shape. In general, if the order of the model is low (small number of coefficients) this problem is avoided but a too low order is not sufficient to preserve the global shape. Finding the optimal order is not straightforward. Experimentally we chose: 25 TECC, 20 MFCC and 15 LPC to model the envelope on pseudo-stationary sung signals.

3.1.2. Intonation: Global description of a note

As explained above, the singing voice differs from the speech in its musical intention and its production. Most of the sung sounds are voiced and sustained. Because of the mode of production of voice two kinds of variations appear on sustained tones:

Vibrato refers to a periodic modulation of frequency. It is a natural effect of singing voice that can be voluntary enhanced by the singer but exists naturally due to the mechanism of the singing production. When a sung tone is emitted with a vibrato, a range of frequencies (centered on the note frequency) is browsed by the vocal tract. Because of its shape, the vocal tract enhances the resonance of some frequencies. So that, depending on the morphology

of the singer, a frequency modulation is accompanied with a modulation of amplitude. The latter is referred as **tremolo** in music. In addition, when two distinct (and distant) notes are sung in the same breath, the singers pass from one to another by a continuous variation of the frequency. If the interval between the notes is small (smaller than a 3^{rd}) the smooth transition is referred to as **legato**. When the gap is higher, the transition is named **portamento**. Portamento is a singing specific term, when string instruments glides continuously from one pitch to another the transition is named glissando.

To obtained information related to periodic and continuous (resp. vibrato and portamento) of a given note, we propose to parameterize the fundamental frequency of a note with the following model:

$$f(t) = \bar{f} \cdot (d_f(t) + x(t)) + \epsilon(t) \quad (3)$$

Where \bar{f} is the mean of $f(t)$ representing the perceived pitch, $x(t)$ is a periodic modulation of frequency representing the vibrato, and $d_f(t)$ is a continuous variation of the pitch representing the portamento.

The parameters can be computed as follow:

- First, \bar{f} is given by the mean of $f(t)$. In order to get equivalent values for two partials with frequencies harmonically related ($f_p(t) = k \cdot f_q(t)$, $k \in \mathbb{Q}$), the other parameters of (3) are computed on a normalized version of the frequency trajectory: $f(t)/\bar{f}$.
- The quantity $f(t)/\bar{f}$ is low-pass filtered with a cutoff frequency $f_c = 4\text{Hz}$. The result of the filtering process is a curve representing the **relative frequency variation** $d_f(t)$ parameterized with a third-order polynom P_{df} .
- The periodic component $x(t)$ is obtained by subtracting the relative frequency deviation $d_f(t)$ from the relative frequency: $x(t) = f(t)/\bar{f} - d_f(t)$.

The vibrato term $x(n)$ can be written as a periodic modulation characterized by an amplitude (or extent) E , a frequency of modulation (or rate) r and a phase at the origin ϕ_0 :

$$x(t) = E \cdot \cos(2\pi r t + \phi_0) \quad (4)$$

The vibrato parameters (only E and r are of interest) are estimated using classical methods for sinusoidal parameters estimation.

As mentioned earlier, in singing the presence of vibrato implies the presence of an amplitude modulation (tremolo) and we suppose the relation between the two modulation singer-specific. The AM's parameters (amplitude and rate) are estimated using equations (3) and (4) applied on the function of amplitude $a(t)$.

$$a(t) = \bar{a} \cdot (d_a(t) + x(t)) + \epsilon(t) \quad (5)$$

In this case, the term \bar{a} is related to the global loudness (or the dynamic $\{p, mf, f, \dots\}$). The low variation $d_a(t)$ transcribes a possible variation of dynamic (*crescendo* for example). The sinusoidal part $x(t)$ represents the amplitude modulation itself.

In practice, the sinusoidal components (partials) are tracked along the studied sound and the analysis is performed on each partial.

3.1.3. Duality of descriptions

We can resume the features obtained on one note, using these two complementary descriptions as presented in Table 1.

Feature type	Local (see sec.3.1.1)	Global (sec.3.1.2)
Nature of the description:	Local variations (High frequencies)	Overall structural information (Low frequencies)
Analyze performed on:	p stationary portions of the signal (frames)	the whole note
Size of the description:	1 feature matrix per note $X = [\vec{x}_1, \dots, \vec{x}_p]$ where \vec{x}_i (with n coeff) is the feature vector obtained on frame i	1 feature vector per trajectory of frequency analyzed: \vec{x} , either the fundamental f_0 or p' partials analyzed
Dimension	$n \times p_{frame}$	$p_{partials} \times n'$

Table 1: Global and local features extracted on a sung tone

3.2. Learning the specificities of classes

It exists numerous statistical algorithms for pattern recognition. We present in table 2 three classes of algorithm that differ in their approach. All of them perform supervised classification. Next, in the evaluation, we used one algorithm (the one given in example) from each class.

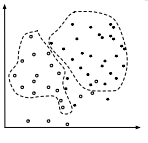
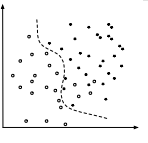
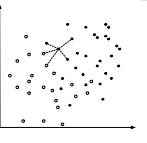
Type	Generative	Discriminative	Instance-based
General Idea			
Principle	Build one model per class	Learn the boundaries between the classes	Compare items
Example	Gaussian Mixture Model GMM	Support Vector Machine SVM	k-Nearest Neighbors kNN
Classify new pattern	Likelihood for each class	Affinity to the margins	Distance to the closest neighbors
Output	Posterior probability	Distance	Distance

Table 2: Different approaches to classify patterns

As shown on the last row of table 2 the outputs of classifiers can have different type and range in different intervals. We can consider, without lost of generality, that all the outputs have values in the interval $[0, 1]$ and represent a (pseudo) posterior probability that an item belong to one class. The transformation of classifier outputs into pseudo-posterior probability can be done using the *softmax* method proposed in [21].

Using the features and the classifier presented above we suggest to identify singer by the two types of information. In the next section we detail the method use for the combination.

3.3. Combination method

The proposed approach is a multi-stage classification method that reduces at each step the set of possible classes until a reduced set of classes remains possible. Then the membership measurements of all classifiers for each remaining class are analyzed to take a final decision.

The approach proposed here is especially adapted to:

- Combine classifications with different levels of performance.
- Solve problems involving a large number of class with no hierarchical organization of the data.
- Combine a low number of representations (when a cascade classification can not be processed until only a single class remain possible)

We first introduce the notation and then present the framework and discuss the choice of the parameters of the method that will later be applied to combine local and global features.

3.3.1. Notations

- Each pattern z (in our case one note) is assigned to one of the N possible **classes**: $\Omega = \{\omega_1 \dots \omega_N\}$.
- Each pattern can be described using different set of features, the set of available **descriptions** D_i is denoted by $\mathcal{D} = \{D_1 \dots D_L\}$.
- The set of **classifiers** is denoted by $\mathcal{C} = \{C_1 \dots C_M\}$.
- A **system of classification** is composed of one description and one classifier: $S^{(k)} = (D^{(k)}, C^{(k)})$ where $D^{(k)} \in \mathcal{D}$ and $C^{(k)} \in \mathcal{C}$.
- In our problem, the first part of the combination is done using a sequential scheme. For a given pattern z , at each step of the classification, the number of possible classes is reduced. So that, each system works with a specific $\Omega^{(k)}(z)$. If $S^{(k)}(z)$ is performed before $S^{(k+1)}(z)$, thus $\Omega^{(k+1)}(z) \subset \Omega^{(k)}(z) \subset \Omega^{(0)} = \Omega$.
- For a given classification task, all systems of classification are trained using the same data set. Since the pattern representation $D^{(k)}$ of this data-set changes for each k , the **training set** associated with $S^{(k)}$ is denoted by $T^{(k)}$. Finally, we denote by $T_{\downarrow}^{(k)}(z)$ the **training-set reduced to patterns with labels in $\Omega^{(k)}(z)$** .
- In the rest of this section we work with classifiers returning a membership measurement for each class of the problem. The output of such a classifier is denoted $C^{(k)}(z) = M^{(k)}(z) = [m_1^{(k)}(z), \dots, m_N^{(k)}(z)]$
- Working on the combination of classifier outputs we store the decision of the K classifiers for the N given classes in a decision profile matrix (of size $N \times K$) denoted by M

3.3.2. General framework

The idea behind our method is the following: The probability to retrieve the correct class of an unknown pattern increases when the number of possible classes of a given classifier decreases. So that, a classification system with a relative low accuracy can enhance the performance of a system with higher accuracy if the problem

given to the weaker system is simplified by the more accurate system. By “simplified problem” we mean a problem with a smaller number of classes.

The general framework can be summarized as follow:

The algorithm starts with a set of N class $\Omega^{(0)} = \Omega$, two descriptions of the same data set $T^{(1)}$ and $T^{(2)}$ and two classifiers $C^{(1)}$ and $C^{(2)}$. For each pattern z , system $S^{(1)}$ returns a measurement vector $M^{(1)}(z)$. The $N^{(1)}$ most likely classes are retained to form a new class-set $\Omega^{(1)} \subset \Omega^{(0)}$. The training-set used by the second system $S^{(2)}$ is reduced to patterns with classes in $\Omega^{(1)}$. The training-set derived from $T^{(2)}$ is denoted $T_{\downarrow}^{(2)}$. Then, the second classification system $S^{(2)}$ trained on $T_{\downarrow}^{(2)}$ is applied to the unknown pattern z . The process can be iterated as long as:

- The last classifier does not return a single class.
- Another system (either a new description of a new classifier) is still available.

If the method is iterated until only one class remain possible ($N^{(K)} = 1$) then the method works as a decision tree. If the process is stopped when $N^{(K)} > 1$ classes remain possible, then the rule for parallel combination can be applied on the output of the classifiers for the $N^{(K)}$ remaining classes. In opposition, if the class-set is not reduced at each step ($N^{(K)} = N$), then the method is equivalent to a classical parallel scheme of combination.

The method is illustrated in Figure 1.

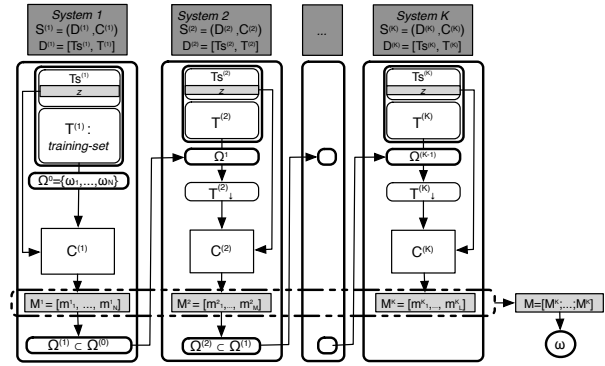


Figure 1: Scheme of the proposed method to combine K systems

We now discuss the choice of the different parameters of the method.

3.3.3. Choice of parameters

Choice of $\Omega^{(k)}$: The number can be simply defined by a relation of type: $N^{(k)} = \frac{N^{(k-1)}}{c^{(k)}}$ where $c^{(k)}$ are defined beforehand. Dynamic rules, as Bayesian information criterion or elbow method, applied on the measurements $m_n^{(k)}$, can be used to define the number of class selected at each step.

Choice of feature space and classifier: There is no restriction on the choice of the descriptions $D^{(k)}$ and the classifiers $S^{(k)}$. Thus for $i \neq j$, the combination system can be set up with $D^{(i)} = D_j$ or $C_i = C_j$. From our experiments the proposed method is still accurate if $S_i = S_j$.

Combination rule: At the end of the sequential stage $N^{(K)}$ classes remain possible. The K vectors of measurements $M^{(k)}(z)$ are reduced to values of classes in $\Omega^{(K)}$ before being combined

into a decision profile matrix M . We suggest to first normalize each column of M on the $N^{(K)}$ remaining classes, and then to apply a *sum-rule* for the reasons explained in [14].

Sequential organization of the $S^{(k)}$: If knowledge on the performances of the K systems is available, we recommend to put the best systems at the top of the iterative process. If all systems have equivalent performances, or if the relative performances cannot be estimated, the systems that require the lowest number of computation should be placed at the end of the process to reduce the cost.

4. EVALUATION OF THE PROPOSED METHOD ON A TASK OF SINGER IDENTIFICATION

We evaluate the proposed method on two distinct sets of data. Both sets are made of isolated notes and we report for each configuration tested the percentage of note assigned to their correct singer. The task is referred to as “closed-set identification” problem (i.e each note belong to one and only one singer of the set).

4.1. Data-set

The two sets are chosen for their complementarities.

The first data set, **LYR**, is composed of recordings made by 17 lyric female singers in laboratory conditions. The full description of this set is given in [22]. For each singer, the same set of tones is available (3 pitches: A5, D5, G4 sung with 3 levels of intensity: p , mf , f and each couple [pitch, intensity] is repeated 3 times). On this set, the task is referred as “closed-set, note-dependent identification” (as text dependent identification).

The second data set, **POP**, has been created by segmenting the vocal track of “pop-rock” songs into sustained notes. For each singer, we work with notes extracted from 3 songs. The task there is referred to as “close-set, note-independent” identification because each singer has a set of notes related to its tessitura. In this set, male and female singers are present.

The two data-sets can be summarized as shown in table 3.

Data-set Name	LYR	POP
Type of voice	Lyric female singers	Rock-Pop singers
Nb of singer	17 (females: F)	18 (8 Males / 10 F)
Nb of sample per singer	27 notes per singer	3 songs per singer segmented into ≈ 50 notes each
Nb of sample per set	$27 \times 17 = 459$ notes	2492 notes
Recordings	Laboratory condition	Personal recording system
Nature	Isolated notes	Notes extracted from songs (in context)
Task	Note-dependent	Note-independent

Table 3: Description of the two data-sets used for the evaluation

4.1.1. Composition of training and testing set

The evaluation is done using supervised machine learning method. Both data-sets are divided into three folds: the training phase is done on the data of 2 folds and the validation is conducted on the

remaining data. Evaluation is done using a 3 folds cross-validation obtained by rotating folds, and for each experiments we report the average accuracy of the 3 experiences.

On LYR, the set of sample available for one singer can be summarized as shown in table 4. To cover the variability of one singer

LYR	p			mf			f		
A5	1	2	3	1	2	3	1	2	3
D5	1	2	3	1	2	3	1	2	3
G4	1	2	3	1	2	3	1	2	3

Table 4: Data available for one singer in LYR

and build more general models, we put into one fold data with all available pitches and intensities. To avoid having too similar data in the training and testing data-set all repetitions of the same note (pitch, intensity) are putted into the same fold. We illustrate in table 4 the repartition of the samples from singer into the 3 folds (where each color represent one fold).

On the POP data-set, each singer uses its own system of recordings and sometimes this system changes from one song to another. To ensure that the identification is performed on the singer identity and not on the song (album effect) we chose to put in one fold all notes extracted from one song. Thus, for each fold evaluation, the singer identity is learned using the notes obtained on two songs and the model obtained is tested on the notes of the remaining song.

4.2. Application of the proposed method

We now evaluate how the singer of a given note is retrieved when using local and global features independently and how the identification is enhanced when local and global features are combined with the method presented in 3.

The combination method is applied for $K=2$ systems of classification where the first system is based on local features and the second one on global features. We thus have:

• **Systems:** $S^{(1)} = (D^{(1)}, C^{(1)})$ and $S^{(2)} = (D^{(2)}, C^{(2)})$

• **Descriptions:** $\mathcal{D} = \{D_1, \dots, D_4\}$ with D_i for $i = 1 \dots 3$ are representations of the data based on local features:

($D_1 \leftarrow \text{TECC}$, $D_2 \leftarrow \text{MFCC}$ and $D_3 \leftarrow \text{LPC}$) and

D_4 is based on global features ($D_4 \leftarrow \text{INTO}$). Thus we have

$$D^{(1)} = D_i \text{ with } i = \{1, 2, 3\} \text{ and } D^{(2)} = D_4$$

. Experimentally we use 25 TECC, 20 MFCC, and 15 LPC.

• **Classifiers:** The available set of classifier is denoted by \mathcal{C} where $\mathcal{C} = \{C_1, C_2, C_3\} = \{SVM, GMM, SVM\}$. All possible configurations are tested for the combination:

$$\forall j, C^{(j)} = C_i \text{ with } i = 1, 2, 3$$

• **Class-set reduction rule:** The number of classes remaining at the end of the first stage is defined dynamically. The membership values are normalized such that their sum is equal to one. The classes that explain 80% of the posterior probabilities are retained to form the new subset of classes of size $N^{(1)}$.

• **Combination rule:** Once the membership measurements for the $N^{(1)}$ remaining classes have been normalized and concatenated to form a decision profile matrix, we apply a “sum-rule” to take the final decision for the reasons explained in [14].

Feature I \ Feature II		TECC			MFCC			LPC	
		SVM (84.97)	kNN (83.22)	GMM (77.56)	SVM (73.42)	kNN (72.55)	GMM (65.36)	SVM (80.61)	kNN (77.78)
Into	SVM (42.48)	89.11	87.8	84.97	79.3	78.21	76.69	86.93	84.97
	GMM (42.70)	86.93	86.27	84.97	79.52	80.39	76.69	86.93	83.88
	kNN (39.22)	87.58	87.8	81.7	78.21	77.12	73.86	84.97	83.88

Table 5: Results of combination method for singer lyric singer identification (LYR)

Feature I \ Feature II		TECC			MFCC			LPC	
		SVM (73.57)	kNN (69.02)	GMM (69.60)	SVM (64.66)	kNN (59.26)	GMM (56.21)	SVM (69.10)	kNN (63.81)
Into	SVM (50.12)	78.97	72.92	74.07	71.37	67.94	56.05	74.85	69.60
	GMM (46.91)	70.72	68.29	72.80	64.97	61.00	60.03	69.92	66.74
	kNN (43.25)	73.92	69.80	71.99	67.01	63.70	61.81	70.41	65.97

Table 6: Results of combination method for singer pop-rock singer identification (POP)

4.3. Results

We present in table 5 the results obtained in the LYR data-set and in table 6 the results obtained on POP data-set.

For both tables, the different configurations of $S^{(1)}$ are presented in the first row and the configurations of $S^{(2)}$ in the first column. The number into bracket placed beside the name of each classifier indicates the accuracy of the system when a single classification is applied. Finally, the accuracies of the combined classifications are reported at the intersection of the two systems used.

The task evaluated here is challenging since only a short segment (a note of few seconds length) is used to recognize the singer. We comment first results with a single type of feature and then comment the results obtained with the combination.

4.3.1. Results on single classification

Single classification with timbre-based features

From the results obtained with TECC, MFCC and LPC (first row of each table) we can deduce that timbre-based features are rather appropriate to describe voice on acapella recordings. However, we remark that results obtained on LYR are much better than results obtained with POP. In LYR, all samples have been recorded in the same ideal conditions (same mic, room) so that we can ensure that the spectral envelopes of these sounds are clearly conveying information on the vocal tract of singers. Probably the results on POP are affected by the “album-effect”. We have also evaluated the performance of classification on POP when the singer models are learned on 2 thirds of each song and the validation is done on the remaining data. With a 3 folds cross-validation the average accuracy obtained is equal to 96%.

For all experiments, the TECC outperform the MFCC and LPC. In both cases, the best result is obtained when working with TECC and SVM. SVM seems to perform better than other classifiers. Even if it is not possible to ensure that each system has been optimized (transformation of the feature space, choice of the classifier parameters, etc.) we see from all these experiments that it is not possible to retrieve the singer with these features even when the task is done on acapella recordings.

Single classification with intonation-based features

Intonative features have not been yet used to singer or instrument recognition. These features can somehow find an equivalent in the prosodic features used in speaker identification. On both data-sets the classifications obtained with INTO features show a relatively good accuracy. We remind that a random classification would

have an accuracy $\approx 5\%$ when working with 17 or 18 singers. The better results obtained on POP can be easily explained. All singers in LYR have a pretty similar technique, and all their vibratos look and sound pretty similar. We do not have any information on the technique of the singers in POP but by comparing the spectrograms (and the partials) of POP singers we can see that the variety in vibrato technique is much larger. The vibrato of lyric singers generally has a large extent and it very regular but the vibrato is definitely present in pop-rock type of voices. From this experiments, we can conclude that expressive elements such as vibrato, tremolo and portamento are singer-specific. Contrary to timbre-based features, intonative features should not be affected by the “album-effect”. Theoretically, the correlation between the amplitude and the frequency modulation should remain constant across the different songs of the singer. According to the analysis done on vibrato rate we can also ensure that the rate of singers’ vibrato does not vary much between different songs of a singer (even if the songs have different tempo or mood).

The capacity to discriminate the classes of each feature composing INTO have been studied using the IRMSFP algorithm proposed in [16]. On the two data-sets the vibrato rate, the tremolo rate and the vibrato extent are the most discriminative features. In POP, the coefficients of the polynomial, representing the portamento, are also of importance. This is mainly due to the fact that the notes composing POP have been extracted from full song, and in many cases the segment analyzed contain note transition.

4.3.2. Results of the combination

In most of cases, combining local and global features with the proposed approach increases the identification accuracy. In average (over all experiments per data-set), a gain of 6.23% and 4.48% is obtained on LYR and POP respectively. In practice, the higher the accuracy of one system is, the more difficult will be to improve the performance by combining a system with a lower accuracy. The gap between the different systems performance is greatly reduced with the double classification. For example, if we consider the results on LYR obtained with a single classification based on LPC ($S^{(1)}, D^{(1)} = \text{LPC}$), the variance of the results obtained with any classifiers is equal to 15 ($\forall C_i, \sigma(\text{Acc}(S^{(1)})) = 15$). When the classifications based on LPC are combined with INTO features, the variance of the results is reduced to 3.44 ($\forall S^{(2)}, \sigma(\text{Acc}(S^{(1)} \cap S^{(2)})) = 3.44$).

This method of combination has been developed because no one of the traditional methods provides an amelioration of the performance already obtained with timbre-based features.

5. CONCLUSION

In this paper we have proposed a new method to identify singer using isolated notes. In the proposed method, the notes to be classified are described using local and global features representing respectively the spectral envelope and some expressive attributes specific to singing. Local descriptors such as MFCC and LPC have been previously used for this kind of experiment but we suggest to transcribe the spectral envelope with a new set of coefficients derived from the true envelope. This new set of coefficients (TECC) show better performances than coefficients traditionally used to transcribe timbre, at least for this task. In general, especially in very clean signals of singing, they perform good classification. In the case studied especially when the classes are learned with SVM. In addition, the set of global features (INTO), previously used to detect the presence of voice within songs, have been proved to be useful to characterize singer identity. They do not obtained results as good as results of timbre-based features but they have the real advantage of being completely orthogonal to the latter. In practice, it is not straightforward to find improve a good classification by introducing information yielding to a poorer classification performance. We have proposed a methods based on the idea that a system of classification with a relative low accuracy can be employed to enhance the classification returned by a stronger system if the problem given to the weaker system is simplified by the best of the two systems. In other words, for a given note, the best system is asked to deduce a subset of possible classes (as small as possible and which still contains the true class) then the second system is asked to perform the classification on the reduced set of classes. Finally, the membership measurements of the reduced set of classes returned by the two classifiers are analyzed to take the final decision. This combination method appear to be efficient for this task since the results obtained by this combination are always better than the results obtained using a single classification.

6. ACKNOWLEDGMENTS

This work was partly supported by the “Quaero” Program funded by Oseo French agency for innovation and by the bourse Dorety Leet/AFFDU. We like to thank C. Dromey for giving us the LYR data-set.

7. REFERENCES

- [1] G. Fant, “The source filter concept in voice production,” in *IV FASE Symposium on Acoustics and Speech, Venezia*, 1981.
- [2] L. Regnier and G. Peeters, “Partial clustering using a time-varying frequency model for singing voice detection,” in *ICASSP*. IEEE, 2010, pp. 441–444.
- [3] B. Whitman, G. Flake, and S. Lawrence, “Artist detection in music with minnowmatch,” *Neural Networks for Signal Processing XI, 2001. Proceedings of the 2001 IEEE Signal Processing Society Workshop*, pp. 559–568, 2001.
- [4] Y.E. Kim and B. Whitman, “Singer identification in popular music recordings using voice coding features,” in *ISMIR*, 2002, pp. 164–169.
- [5] H. Fujihara, T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno, “Singer identification based on accompaniment sound reduction and reliable frame selection,” *Proc. ISMIR*, pp. 329–336, 2005.
- [6] A. Mesaros, T. Virtanen, and A. Klapuri, “Singer identification in polyphonic music using vocal separation and pattern recognition methods,” in *ISMIR*, 2007, pp. 375–378.
- [7] NC Maddage, C. Xu, and Y. Wang, “Singer identification based on vocal and instrumental models,” *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2, 2004.
- [8] W.H. Tsai and H.M. Wang, “Automatic singer recognition of popular music recordings via estimation and modeling of solo vocal signals,” *Audio, Speech and Language Processing, IEEE Transactions on*, vol. 14, no. 1, pp. 330–341, 2006.
- [9] A. Berenzweig, D.P.W. Ellis, and S. Lawrence, “Using voice segments to improve artist classification of music,” *AES 22nd International Conference*, 2002.
- [10] Y.E. Kim, D.S. Williamson, and S. Pilli, “Towards quantifying the album effect in artist identification,” in *ISMIR*. Citeseer, 2006, pp. 393–394.
- [11] M.A. Bartsch, *Automatic singer identification in polyphonic music*, Ph.D. thesis, The University of Michigan, 2004.
- [12] L. Xu, A. Krzyzak, and C.Y. Suen, “Methods of combining multiple classifiers and their applications to handwriting recognition,” *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 3, pp. 418–435, 2002.
- [13] L.I. Kuncheva, *Combining pattern classifiers: methods and algorithms*, Wiley-Interscience, 2004.
- [14] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, “On combining classifiers,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 3, pp. 226–239, 2002.
- [15] C.L. Liu, “Classifier combination based on confidence transformation,” *Pattern Recognition*, vol. 38, no. 1, pp. 11–28, 2005.
- [16] G. Peeters, “Automatic classification of large musical instrument databases using hierarchical classifiers with inertia ratio maximization,” *115th AES convention, New York, USA, October*, 2003.
- [17] P. Zhang, T.D. Bui, and C.Y. Suen, “A novel cascade ensemble classifier system with a high recognition performance on handwritten digits,” *Pattern Recognition*, vol. 40, no. 12, pp. 3415–3429, 2007.
- [18] T.E. Senator, “Multi-stage classification,” in *Data Mining, Fifth IEEE International Conference on*. IEEE, 2005, p. 8.
- [19] S. Imai and Y. Abe, “Spectral envelope extraction by improved cepstral method,” *Electron. and Commun. in Japan*, vol. 62, pp. 10–17, 1979.
- [20] A. Röbel, “Efficients spectral envelope estimation and its application to pitch shifting and envelope preservation,” in *DAFx’x*, 2005.
- [21] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern classification*, vol. 2, Wiley, 2001.
- [22] C Dromey, N Carter, and A Hopkin, “Vibrato rate adjustment,” *Journal of Voice*, vol. 17, no. 2, pp. 168–178, 2003.

ENHANCED BEAT TRACKING WITH CONTEXT-AWARE NEURAL NETWORKS

Sebastian Böck, Markus Schedl

Department of Computational Perception
Johannes Kepler University, Linz
Austria
sebastian.boeck@jku.at

ABSTRACT

We present two new beat tracking algorithms based on the autocorrelation analysis, which showed state-of-the-art performance in the MIREX 2010 beat tracking contest. Unlike the traditional approach of processing a list of onsets, we propose to use a bidirectional Long Short-Term Memory recurrent neural network to perform a frame by frame beat classification of the signal. As inputs to the network the spectral features of the audio signal and their relative differences are used. The network transforms the signal directly into a beat activation function. An autocorrelation function is then used to determine the predominant tempo to eliminate the erroneously detected - or complement the missing - beats. The first algorithm is tuned for music with constant tempo, whereas the second algorithm is further capable to follow changes in tempo and time signature.

1. INTRODUCTION

For humans, tracking the beat is an almost natural task. We tap our foot or nod our head to the beat of the music. Even if the beat changes, humans can follow it almost instantaneously. Nonetheless, for machines the task of beat tracking is much harder, especially when dealing with varying tempi, as the numerous publications by different authors on this subject suggest.

Locating the beats precisely opens new possibilities for a wide range of music applications, such as automatic manipulation of rhythm, time-stretching of audio loops, beat accurate automatic DJ mixing or self-adapting digital audio effects. Beats are also crucial for analyzing the rhythmic structure, and the genre of songs. In addition they help identifying cover songs or estimating the similarity of music pieces.

The remainder of this paper is structured as follows: Section 2 gives a short overview over existing methods for beat tracking. Section 3 briefly introduces the concept and different types of neural networks with a special emphasis on bidirectional Long Short-Term Memory recurrent neural networks, which are used in the proposed algorithms. Section 4 details all aspects of the newly proposed beat tracking algorithms. Results and discussion are given in Section 5 and the final section presents conclusions and an outlook to further works.

2. RELATED WORK

Most methods for beat tracking of audio signals have a working scheme like the one shown in Figure 1. After extracting features from the audio signal, they try to determine the periodicity of the signal (the tempo) and the phase of the periodic signal (the beat

locations). The features can be for example onsets, chord changes, amplitude envelopes, or spectral features. The choice of a particular feature mostly depends on the subsequent periodicity estimation and phase detection stages. For periodicity estimation, autocorrelation, comb filter, histogram, and multiple agent based induction methods are widely used. Some methods also produce phase information during periodicity estimation, and therefore do not need a phase detection stage to determine the exact position of the beat pulses. [1] gives a good overview on the subject.

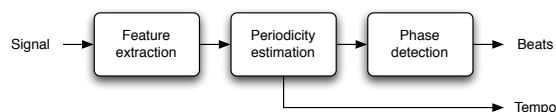


Figure 1: Basic workflow of traditional beat tracking methods.

Most of today's top performing beat tracking algorithms rely on onsets as features [2, 3, 4]. Since music signals contain much more onsets than beats, additional processing is needed to locate the beats within the onsets. By transferring this determination of beats into a neural network, less complex post-processing is needed to achieve comparable or better results.

3. NEURAL NETWORKS

Neural networks have been around for decades and are successfully used for all kind of machine learning tasks.

The most basic approach is the *multilayer perceptron* (MLP) forming a *feed forward neural network* (FNN). It has a minimum of three layers where the input values are fed through one or more hidden layers consisting of neurons with non-linear activation functions. The output values of the last hidden layer are finally gathered in the output nodes. This type of network is a strictly causal one, where the output is calculated directly from the input values.

If cyclic connections in the hidden layers are allowed *recurrent neural networks* (RNN) are formed. They are theoretically able to remember any past value. In practice however, RNNs suffer from the vanishing gradient problem, i.e. input values decay or blow up exponentially over time.

In [5] a new method called *Long Short-Term Memory* (LSTM) is introduced to overcome this problem. Each LSTM block (depicted in Figure 2) has a recurrent connection with weight 1.0 which enables the block to act as a memory cell. Input, output, and forget gates control the content of the memory cell through multiplicative units and are connected to other neurons as usual.

If LSTM blocks are used, the network has access to all previous input values.

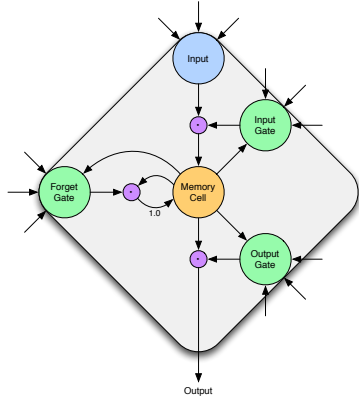


Figure 2: LSTM block with memory cell

If not only the past, but also the future context of the input is necessary to determine the output, a number of different strategies can be applied. One is to add a fixed time window to the input, another solution is to add a delay between the input values and the output targets. Both measures have their downsides as they either increase the input vector size considerably or the input values and output targets are displaced from each other.

Bidirectional recurrent neural networks (BRNN) offer a more elegant solution to the problem by doubling the number of hidden layers. The input values to the newly created set of hidden layers are presented to the network in reverse temporal order. This offers the advantage that the network not only has access to past input values but can also 'look into the future'.

If bidirectional recurrent networks are used in conjunction with LSTM neurons, a *bidirectional Long Short-Term Memory* (BLSTM) recurrent neural network is build. It has the ability to model any temporal context around a given input value. BLSTM networks performed very well in areas like phoneme and handwriting recognition and are described more detailed in [6].

4. ALGORITHM DESCRIPTION

This section describes our algorithm for beat detection in audio signals. It is based on bidirectional Long Short-Term Memory (BLSTM) recurrent neural networks. Due to their ability to model the temporal context of the input data [6], they perfectly fit into the domain of beat detection. Inspired by the good results for musical onset detection [7], the approach of this work is used as a basis and extended to suit the needs for audio beat detection by modifying the input representation and adding an advanced peak detection stage.

Figure 3 shows the basic signal flow of the proposed system. The audio data is transformed to the frequency domain via three parallel Short Time Fourier Transforms (STFT) with different window lengths. The obtained magnitude spectra and their first order differences are used as inputs to the BLSTM network, which produces a beat activation function. In the peak detection stage, first the periodicity within this activation function is detected with the autocorrelation function to determine the most dominant tempo.

The beats are then aligned according to the previously computed beat interval. We propose two different peak detection algorithms, one tuned for music with constant tempo and beats (*BeatDetector*) and a second one which is able to track tempo changes (*BeatTracker*). The individual blocks are described in more detail in the following sections.

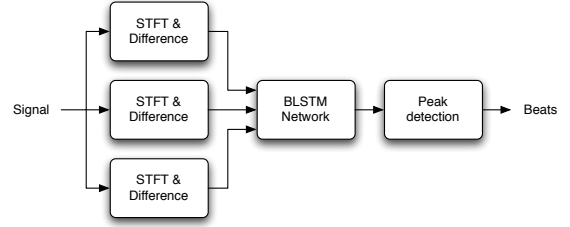


Figure 3: Basic signal flow of the presented beat detector / tracker

4.1. Feature Extraction

As input, the raw pulse code modulated (PCM) audio signal with a sampling rate of $f_s = 44.1$ kHz is used. To reduce the computational complexity, stereo signals are converted to a monaural signal by averaging both channels. The discrete input audio signal $x(n)$ is segmented into overlapping frames of W samples length. The windows with lengths of 23.2 ms, 46.4 ms, and 92.8 ms (1024, 2048, and 4096 samples respectively) are sampled every 10 ms, resulting in a frame rate $f_r = 100$ fps. A standard Hamming window $w(l)$ of the same length is applied to the frames before the STFT is used to compute the complex spectrogram $X(n, k)$

$$X(n, k) = \sum_{l=-\frac{W}{2}}^{\frac{W}{2}-1} w(l) \cdot x(l + nh) \cdot e^{-2\pi jlk/W} \quad (1)$$

with n being the frame index, k the frequency bin index, and h the hop size or time shift in samples between adjacent frames. The complex spectrogram is converted to the power spectrogram $S(n, k)$ by omitting the phase portion of the spectrogram by:

$$S(n, k) = |X(n, k)|^2 \quad (2)$$

Psychoacoustic knowledge is used to reduce the dimensionality of the resulting magnitude spectra. To this end, a filterbank with 20 triangular filters located equidistantly on the Mel scale is used to transform the spectrogram $S(n, k)$ to the Mel spectrogram $M(n, m)$. To better match the human perception of loudness, a logarithmic representation is chosen (cf. Figure 4(a)):

$$M(n, m) = \log \left(S(n, k) \cdot F(m, k)^T + 1.0 \right) \quad (3)$$

If large window lengths are used for the STFT, the raise of the magnitude values in the spectrogram occurs early compared to the actual beat location (cf. Figure 4(b)). Instead of calculating the simple positive first order difference as in [7], a more advanced method is used to overcome this displacement of the actual beat locations compared to the positive first order difference. First a median spectrogram $M_{median}(n, m)$ is obtained according to

$$M_{median}(n, m) = \text{median}\{M(n - l^*, m), \dots, M(n, m)\} \quad (4)$$

with l^* being the length for which the median is calculated. This length depends on the used window size W for the STFT, and is computed as: $l^* = \lfloor W/100 \rfloor$. Both the use of the median and the length of the window were empirically determined during preliminary studies. The positive first order median difference $D^+(n, m)$ is then calculated as

$$D^+(n, m) = H(M(n, m) - M_{median}(n, m)) \quad (5)$$

with $H(x)$ being the half-wave rectifier function $H(x) = \frac{x+|x|}{2}$ (cf. Figure 4(c)). Using only the positive differences as additional inputs to the neural network gave better performance than omitting the differences at all or including both the positive and negative values.

4.2. Neural Network

For the neural network stage, a bidirectional recurrent neural network with LSTM units is used. As inputs to the neural network three logarithmic Mel-spectrograms $M_{23}(n, m)$, $M_{46}(n, m)$ and $M_{93}(n, m)$ (computed with window sizes of 23.2 ms, 46.4 ms, and 92.8 ms, respectively) and their corresponding positive first order median differences $D_{23}^+(n, m)$, $D_{46}^+(n, m)$, and $D_{93}^+(n, m)$ are used, resulting in 120 input units. The fully connected network has three hidden layers in each direction, with 25 LSTM units each (6 layers with 150 units in total). The output layer has two units, representing the two classes ‘beat’ and ‘no beat’. Thus the network can be trained as a classifier with the cross entropy error function. The outputs use the softmax activation function, i.e., the output of each unit is mapped to the range $[0, 1]$ and their sum is always 1. The output nodes thus represent the probabilities for the two classes.

4.2.1. Network Training

The network is trained as a classifier with supervised learning and early stopping. The used training set consists of 88 audio excerpts taken from the ISMIR 2004 tempo induction contest¹ (also known as the “Ballroom set”) with lengths of 10 seconds each, the 26 training and bonus files from the MIREX 2006 beat tracking contest² with lengths of 30 seconds, and 6 musical pieces of the set introduced by Bello in [8] with lengths from 3 to 15 seconds. Each musical piece is manually beat annotated, marking every quarter note in case of time signature with a denominator of four (i.e., 2/4, 3/4, and 4/4), and the eighth note for all pieces (or parts of pieces) with a time signature of 5/8 or 7/8. The 120 files have a total length of 28.5 minutes and 3,595 annotated beats.

Each audio sequence is preprocessed as described above and presented to the network for learning. The network weights are initialized with random values following a Gaussian distribution with mean 0 and standard deviation 0.1. Standard gradient descent with backpropagation of the errors is used to train the network. To prevent over-fitting, the performance is evaluated after each training iteration on a separate validation set (a 15% randomly chosen disjoint part of the training set). If no improvement is observed for

20 epochs, the training is stopped and the network state with the best performance on the validation set is used onwards.

4.2.2. Network Testing

Since the network weights were initialized randomly, five different networks were trained on different sets of the training data. The beat activation functions of the ‘beat’ output nodes are then averaged and used as input to the following stage (cf. Figure 4(d)). For the evaluation the preprocessed music excerpts are presented to these five previously trained networks.

4.3. Peak Detection

The averaged beat activation function (cf. Figure 4(d)) gives the probability of a beat at each frame. Similar to [7], the function could be used directly to determine the beats by applying a simple threshold. However, a more sophisticated algorithm for peak picking is applied here. It is able to reduce the relatively high number of false positives and negatives even further. This method yields an F-measure value of 0.88 for a 5-fold cross validation on the complete training set, compared to 0.81 achieved using a simple threshold.

If constant tempo is assumed for (a part of) the musical piece, the predominant tempo can be used to eliminate false positive beats, or complement missing false negative ones. The two different proposed peak detection techniques differ only in the length for which a constant tempo is assumed. The *BeatDetector* assumes a constant tempo throughout the whole musical piece, whereas the *BeatTracker* considers only a moving window which covers the next 6 seconds. This modification enables the *BeatTracker* to follow tempo changes.

4.3.1. Autocorrelation Function

Both proposed algorithms first determine the tempo for the musical piece. The *BeatDetector* uses the entire input signal for calculation, whereas the *BeatTracker* only uses the next 6 seconds relative to the actual starting point. The most dominant beat interval of this segment is used to estimate the tempo. The autocorrelation function (ACF) is calculated on the beat activation function $a_b(n)$ as follows:

$$A(\tau) = \sum_n a_b(n + \tau) \cdot a_b(n) \quad (6)$$

The algorithm constrains the possible tempo range of the audio signal from $T_{min} = 40$ to $T_{max} = 220$ given in beats per minute. Thus only values of $A(\tau)$ corresponding to the range from $\tau_{min} = 273$ ms to $\tau_{max} = 1.5$ s are used for calculation. Since music tends to slightly vary in tempo and beats sometimes occur early or late relative to the absolute position of the dominant tempo, the resulting inter beat intervals vary as well. Therefore a smoothing function s is applied to the result of the autocorrelation function $A(\tau)$. A Hamming window with a size of $\tau_t = 150$ ms is used. The size of this window is not crucial, as long as it is wide enough to cover all possible interval fluctuations and remains shorter than the smallest delay τ_{min} used for the autocorrelation. This results in the smoothed autocorrelation function $A^*(\tau)$:

$$A^*(\tau) = A(\tau) \star s(\tau_t) \quad (7)$$

¹<http://mtg.upf.edu/ismir2004/contest/tempoContest/node5.html>

²http://www.music-ir.org/mirex/wiki/2006:Audio_Beat_Tracking

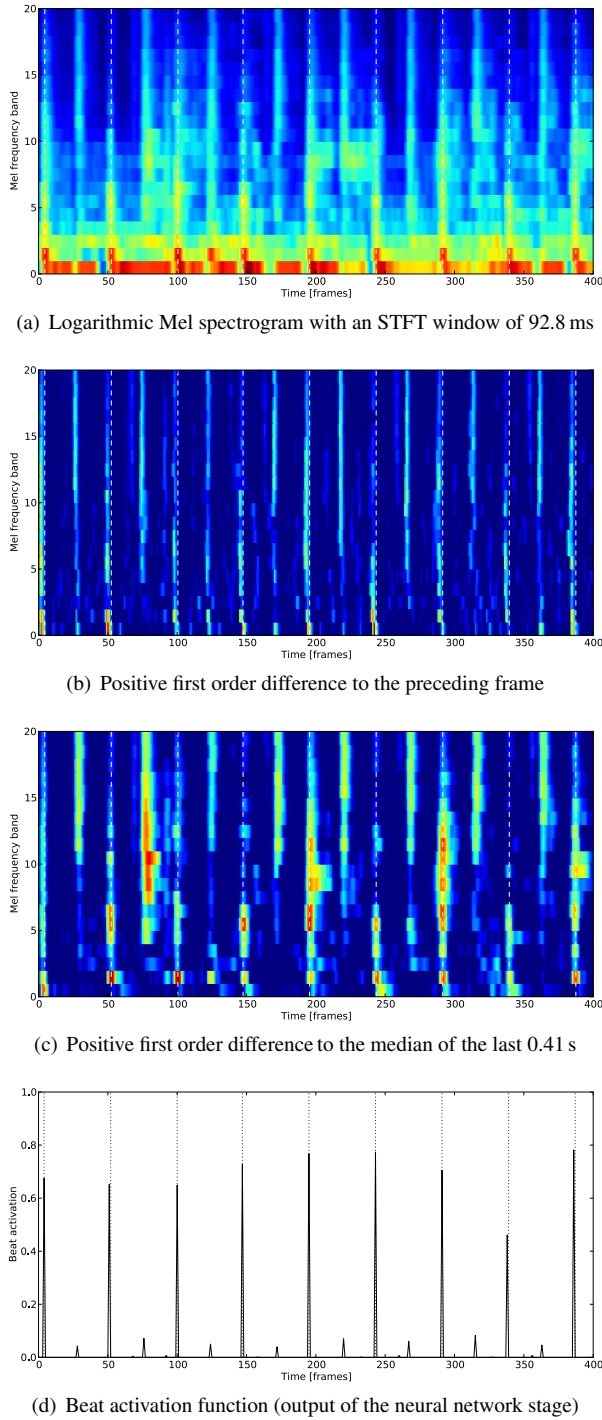


Figure 4: Evolution of the signal through the processing steps of the algorithm. It shows a 4 s excerpt from ‘Basement Jaxx - Rendez-Vu’. Beat positions are marked with dashed/dotted vertical lines.

4.3.2. Beat Phase Detection

The dominant tempo T corresponds to the highest peak in the smoothed autocorrelation function $A^*(\tau)$ at index τ^* . This delay τ^* is used as the beat interval i . The phase of the beat p^* is computed as the highest value of the beat activation function’s sum at the possible beat positions for the given interval i :

$$p^* = \max_{p=0 \dots i} \sum_k a_b(p + k \cdot i) \quad (8)$$

4.3.3. Peak Picking

Finally, the beats are represented by the local maxima of the beat activation function. Thus, we use a standard peak search around the locations given by $n_k = p^* + k \cdot i$ calculated with the previously determined p^* . To allow for small timing fluctuations, a deviation factor $d = 0.1 \cdot i$ is introduced and the final beat function $b(n)$ is given by:

$$b(n) = \begin{cases} 1 & \text{for } a_b(n_k - d) \leq a_b(n_k) \leq a_b(n_k + d) \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

The *BeatDetector* determines all beats in this manner. The *BeatTracker* only detects the next beat and moves the beginning of the lookahead window to that beat. Then the dominant tempo estimation and all consecutive steps (Section 4.3.1 to 4.3.3) are performed on the new section of the beat activation function.

5. EVALUATION

Beat tracking performance was evaluated during the MIREX 2010 beat tracking contest with two different datasets³. The first set, the McKinney collection (MCK set), has rather stable tempo. The second collection (MAZ set) consists of Chopin Mazurkas, which are in 3/4 time signature and contain tempo changes.

Both described algorithms outperformed all other contributions on the MCK set. The *BeatDetector* shows a small overall advantage over the *BeatTracker*. Depending on the used performance measure the relative performance gain compared to the next best algorithm is up to 5.7% (F-measure with a detection window of ± 70 ms), 6.9% (Cemgil: accuracy based on a Gaussian error function with 40 ms std. dev.), 8.2% (Goto: binary decision based on statistical properties of a beat error sequence), and 4.7 (PScore: McKinney’s impulse train cross-correlation method). Table 1 summarizes the results and also includes the best result ever achieved in the MIREX competition by any algorithm as a reference to the state-of-the-art. It can be seen that our *BeatTracker* algorithm performs better or close to it (depending on the used performance measure). This shows the future potential of this approach compared to other signal based ones, given the fact that the actual peak picking algorithm is a rather simple one.

The tempo changes of the MAZ set are the main reason for the *BeatDetector* not performing better (see Table 2), as it assumes a constant tempo throughout the whole musical piece. Nonetheless the algorithm performs still reasonably well. As expected, the more flexible *BeatTracker* performs better and ranks second according to F-measure and first according to Cemgil’s performance

³Evaluation measures described at http://www.music-ir.org/mirex/wiki/2010:Audio_Beat_Tracking#Evaluation_Procedures

MCK Set [%]	F-measure	Cemgil	Goto	PScore
BeatTracker	54.50	41.30	8.87	59.19
BeatDetector	53.16	40.28	22.64	57.73
GP3	50.27	37.21	20.92	56.54
LGG2	49.97	37.68	17.93	54.96
TL2	41.97	29.86	2.50	50.59
NW1	35.56	25.83	5.75	45.67
MRVCC1	25.70	18.34	0.14	38.36
ZTC1	24.64	18.61	0.41	26.07
GPI (2009)	54.80	41.00	22.20	59.00

Table 1: Results for the MIREX 2010 beat tracking evaluation (MCK set). Only the best performing algorithm of other participants are shown; GPI & GP3: Peeters, LGG2: Oliveira et. al., TL2: Lee, NW1: Wack et. al., MRVCC1: Campos et. al., ZTC1: Zhu et. al.

measure. However, the most mentionable aspect is that the neural networks were trained solely on ballroom dance and other kinds of western pop music. Neither a classical piece nor piano music was used for training. Furthermore, only one training example actually contained tempo changes. This suggest that even better performance can be expected when trained on music which has properties similar to the MAZ data set.

MAZ Set [%]	F-measure	Cemgil	Goto	PScore
TL2	68.46	40.42	0.00	72.21
BeatTracker	58.74	51.81	0.00	57.92
MRVCC2	49.26	39.55	0.31	51.22
GP4	48.27	36.72	0.31	50.06
BeatDetector	47.30	38.20	0.00	45.92
LGG2	41.48	30.65	0.00	43.51
NW1	27.59	19.82	0.00	31.35
ZTC1	1.16	0.94	0.00	0.94

Table 2: Results for the MIREX 2010 beat tracking evaluation (MAZ set). Only the best performing algorithm of other participants are shown; TL2: Lee, MRVCC2: Campos et. al., GP4: Peeters, LGG2: Oliveira et. al., NW1: Wack et. al., ZTC1: Zhu et. al.

6. CONCLUSIONS AND FUTURE WORK

This paper presented two novel beat tracking algorithms which perform state-of-the-art although they use a relatively simple and straight forward approach. The *BeatTracker* outperformed all other algorithms in the MIREX 2010 beat tracking contest for the McKinney dataset. Although no classical music was used for training and the training set had less then 3.5 minutes of material with a time signature of 3/4 the new *BeatTracker* performed still reasonably well on the Mazurka test set (all excerpts are in 3/4 time signature). This shows the aptitude of the BLSTM neural network for correctly modeling the temporal context and directly classifying beats. Since the *BeatTracker* shows superior performance over the more simple *BeatDetector* even for musical excerpts with constant tempo, future development will concentrate on this algorithm.

Besides training with a more comprehensive training set, future work should also investigate a possible performance boost by implementing some more advanced beat tracking algorithms in the peak detection stage. Kalman filters [9], particle filters [10], a multiple agents architecture [11] and dynamic programming [2] seem promising choices. Another possibility is the inclusion of other input features which haven proven to be effective for identifying beats [12].

7. ACKNOWLEDGMENTS

This research is supported by the Austrian Science Funds (FWF): P22856-N23.

8. REFERENCES

- [1] F. Gouyon and S. Dixon, "A review of automatic rhythm description systems," *Computer Music Journal*, vol. 29, pp. 34–54, February 2005.
- [2] D. P. W. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.
- [3] M. Davies and M. Plumbley, "Context-dependent beat tracking of musical audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1009–1020, March 2007.
- [4] G. Peeters, "Beat-marker location using a probabilistic framework and linear discriminant analysis," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, September 2009.
- [5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [6] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Ph.D. thesis, Technische Universität München, 2008.
- [7] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long short-term memory neural networks," in *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, August 2010, pp. 589–594.
- [8] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1035–1047, September 2005.
- [9] Y. Shiu, N. Cho, Chang P.-C., and C.-C. Kuo, "Robust on-line beat tracking with kalman filtering and probabilistic data association (kf-pda)," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1369–1377, August 2008.
- [10] S. Hainsworth and M. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP J. Appl. Signal Process.*, vol. 2004, pp. 2385–2395, January 2004.
- [11] S. Dixon, "Evaluation of the Audio Beat Tracking System BeatRoot," *Journal of New Music Research*, vol. 36, no. 1, pp. 39–50, 2007.
- [12] F. Gouyon, G. Widmer, X. Serra, and A. Flexer, "Acoustic cues to beat induction: A machine learning perspective," *Music Perception. UCPress*, vol. 24, pp. 177–188, 2006.

ON THE USE OF PERCEPTUAL PROPERTIES FOR MELODY ESTIMATION

Wei-Hsiang Liao and Alvin W. Y. Su

Dep. of Computer Science and Information Engineering
National Cheng-Kung University, Tainan, Taiwan
whsng.liao@gmail.com, alvinsu@mail.ncku.edu.tw

Chunghsin Yeh and Axel Roebel

Analysis/Synthesis team
IRCAM/CNRS-STMS Paris, France
cyeh@ircam.fr, roebel@ircam.fr

ABSTRACT

This paper is about the use of perceptual principles for melody estimation. The melody stream is understood as generated by the most dominant source. Since the source with the strongest energy may not be perceptually the most dominant one, it is proposed to study the perceptual properties for melody estimation: loudness, masking effect and timbre similarity. The related criteria are integrated into a melody estimation system and their respective contributions are evaluated. The effectiveness of these perceptual criteria is confirmed by the evaluation results using more than one hundred excerpts of music recordings.

1. INTRODUCTION

Auditory scene analysis of music signals is an ongoing active research in recent years as encouraging results continue to explore various applications in the field of digital audio effects (DAFx) and music information retrieval (MIR) [1]. Among the harmonic sources present in the music scene, the “melody” source usually forms perceptually and musically the most dominant stream [2] [3] [4]. The problem of melody estimation is difficult because it requires not only low-level information about sound signals but also high-level information about perception of music. In this article, we define the melody estimation problem as the estimation of the fundamental frequency (F0) of the most dominant source stream. Since the source with the strongest energy may not be perceptually the most dominant one, our study will make use of perceptual properties and evaluate their effectiveness.

In addition to the perceptual grouping cues of harmonic sounds in auditory scene analysis [5], many of the existing methods for melody estimation further make use of other perceptual properties such as loudness [6] [7], masking [8], timbre similarity [6] [9] [10] and auditory filters [3] [11] [12]. If one looks at the evaluation results of the MIREX (Music Information Retrieval Evaluation eXchange) campaign for the “Audio Melody Estimation” task, the systems that make use of these perceptual properties seem to show certain advantages in performance. In fact, the perceptually-motivated system proposed by Dressler [13, 14, 9, 15] always ranks the top [16]. Although important details of perceptual criteria are missing in her descriptions, it is nevertheless reasonable to assume that the key problem of melody estimation is related to perceptual criteria. In this study, we propose to evaluate the following perceptual criteria: loudness, masking, and timbre similarity within the proposed melody estimation system. The auditory filters and other multi-resolution analysis methods are not explored here because we believe that the melody source stream is usually significantly present in the mid-frequency range and a fixed resolution of STFT (short-time Fourier transform) can thus be sufficiently adapted.

The proposed system consists mainly of two parts: candidate selection and tracking. As the salience of an F0 candidate is derived from the the dominant peaks that are harmonically matched, we propose to compare perceptually-motivated criteria with low-level signal features for dominant peak selection. Similarly, candidate scoring based on perceptual criteria is also evaluated to reveal how a correct candidate can be more favored than others. Based on the algorithm previously proposed in [17], a tracking algorithm dedicated to melody estimation is developed to determine the coherent source stream with an optimal trade-off among candidate score, smoothness of frequency trajectory and spectral envelope similarity.

The paper is organized as follows: In Section 2, we present the methods for dominant peak selection and candidate scoring. In Section 3, the components of the tracking system is detailed. In Section 4, the effectiveness of the perceptual criteria are evaluated and the performance of the proposed system is compared to the state-of-the-art systems. Finally, conclusions are drawn and future works are proposed.

2. CANDIDATE EXTRACTION

Extraction of compact F0 candidates from polyphonic signals is not an easy task because concurrent sources interfere with each other and spectral components from different sources may form reasonable F0 hypotheses [18]. Although a proper multiple-F0 estimation allows proper treatment of overlapping partials, a simpler scheme shall meet our needs for melody estimation.

Under the assumption that the melody stream is generated by the most dominant source, the interference from other sources has less impact on its spectral components. The remaining problem is then to avoid extracting subharmonic F0 candidates that are supported by the combination of spectral components from different sources. They appear to be very competitive to the correct F0 and are very likely to cause octave errors. Since the target source is assumed to be dominant, its harmonic components should be present as dominant spectral peaks. By means of selecting the dominant peaks, we can avoid excessive spurious candidates and efficiently establish a compact set of F0 hypotheses with reliable salience.

2.1. Peak Selection

We propose four peak selection methods. The first two are based on loudness weighting and masking effects respectively to select *perceptually dominant* peaks, and the other two are based on cepstral envelope and noise envelope respectively to select *energy dominant* peaks.

Select by Loudness

It is known that the relative energy of the spectral components one measures is very different from the relative loudness one perceives [19]. Since calculating the loudness for complex sound is not straightforward, a common approach is to apply proper spectral weighting by a selected equal-loudness contour to imitate the perceptual dominance of spectral components. Accordingly, we weight the spectrum X with a frequency dependent equal-loudness curve L to obtain the **loudness spectrum** X_L :

$$X_L(k) = \frac{X(k)}{L(k)}; \quad (1)$$

where k is the frequency bin. We choose the equal-loudness curve proposed by Fletcher and Munson [20] measuring at 0dB SPL (sound pressure level) for L :

$$20 \log_{10} L(k) = 3.64 \cdot f_k^{-0.8} - 6.5 \cdot e^{-0.6 \cdot (f_k - 3.3)^2} + (10^{-3}) \cdot f_k^4 \quad (2)$$

where the frequency f_k in “kHz” is converted from the respective frequency bin k . Then, we select the peaks that are not smaller than δ_L dB of the maximum of X_L (see Fig. 1(a)).

Select by Masking Curve

The masking effect depicts how a tone can mask its neighboring components across critical bands, which can be represented by the spreading function (on dB scale) [21]

$$S_f(i, j) = 15.81 + 7.5((i - j) + 0.474) - 17.5(1 + ((i - j) + 0.474)^2)^{0.5} \quad (3)$$

where i is the bark frequency of the masking signal, and j is the bark frequency of the masked signal. The formula of converting frequency f_k from “kHz” to the bark scale is [22]:

$$B(f_k) = 13 \cdot \arctan(0.76 \cdot f_k) + 3.5 \cdot \arctan\left(\frac{f_k}{7.5}\right)^2 \quad (4)$$

The strength of masking of a peak is not only determined by the magnitude of the peak, but also related to its being *tonal* or *noisy*. We follow the MPEG’s standard to classify a peak [23]: If a peak is 7dB higher than its neighboring component, it is considered tonal. Otherwise, it is considered noisy. Accordingly, the mask contributed by a peak is thus (on dB scale):

$$M(i, j) = S_f(i, j) - (14.5 + i) \cdot \alpha - 5.5 \cdot (1 - \alpha) \quad (5)$$

(*tonal* : $\alpha = 1$, *noisy* : $\alpha = 0$)

By means of selecting the maximal mask overlaying at each bin, the masking curve X_m is constructed:

$$20 \log_{10} X_m(k) = \max\{M(i, B(f_k))\}, \forall i \in I \quad (6)$$

where I is the set of all peaks. The peaks which are larger than the masking curve are selected (see Fig. 1(b)).

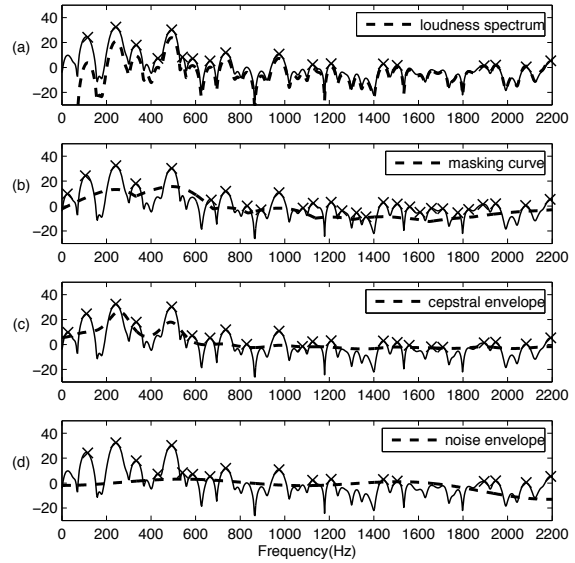


Figure 1: Dominant peak selection by (a) loudness spectrum, (b) masking curve, (c) cepstral envelope, and (d) noise envelope. The original spectrum is plotted as thin solid line and the selected peaks are marked by crosses. The y-axis is the log-amplitude in dB.

Select by Cepstral Envelope

The cepstral envelope is an approximation of the expected log-amplitude of the spectrum [24]. That is, it is a frequency-dependent curve that passes through the mean log-amplitudes at respective frequencies. Accordingly, it is reasonable to assume that the spectral peaks of the most dominant source lie above the cepstral envelope (see Fig. 1(c)). An optional raise of δ_C dB can be used to prevent selection of noise peaks.

Select by Noise Envelope

For the case of polyphonic signals, the cepstral envelope may not give reasonable estimation due to dense distribution of sinusoidal peaks. Besides, it allows some noise peaks to be selected because it passes through the mean of the noise peaks. A solution to these problems is the use of the noise envelope which is the raise of the mean noise level [18]. The proposed noise level estimation makes use of the Rayleigh distribution to model the spectral magnitude distribution of noise and is adaptive in frequency [25]. We raise the mean noise level by δ_N dB as the noise envelope to select dominant peaks (see Fig. 1(d)).

2.2. Candidate Generation and Scoring

Harris suggested locating all groups of pitch harmonics by means of identifying equally spaced spectral peaks on which the salience of a group is built [26]. This method belongs to the **spectral interval** type F0 estimators [27]. For polyphonic signals, however, partials belonging to different sources may form a group of harmonics which results in subharmonic F0s. One way to avoid generating subharmonic F0 candidates is to cast further constraints on the **spectral location** of each partial. Similar to the *inter-peak*

beating method proposed in [18], we present a method for generating F0 candidates from the selected dominant peaks. First, the F0 hypotheses are generated by collecting the spectral intervals between any pair of dominant peaks in the spectrum. Then, the spectral location principle is applied: If the generated hypothesis is not harmonically related to the peaks that support its spectral interval, it is not considered a reasonable candidate. Due to the overlapping partials, frequencies of the peaks are not sufficiently precise. Thus, a semitone tolerance is allowed for the harmonic matching.

In order to reflect the perceptual dominance of a candidate, we propose to score F0 candidates based on the loudness spectrum X_L (eq. 1): the score of a candidate is the summation of the first $H = 10$ partials in the loudness spectrum. The contribution of a partial is determined by the harmonically matched peak with the largest loudness nearby. The partials not selected as dominant peaks will not contribute to the score.

3. TRACKING BY DYNAMIC PROGRAMMING

Given a sequence of candidates extracted from the spectrogram, we adapt the tracking algorithm proposed in [17] to decode the melody stream. Since the melody stream may not be always the most dominant source at each short-time instant, decoding with the maximal score will not yield the optimal result. Therefore, we propose to integrate an additional criterion, spectral envelope similarity, into the dynamic programming scheme. Following [17], we describe the problem using the hidden Markov model (HMM):

- Hidden state: true melody F0
- Observation: loudness spectrogram
- Emission probability: normalized candidate score
- Transition probability
 - trajectory smoothness: the frequency difference between two connected F0 candidates
 - spectral envelope similarity: the spectral envelope difference between two connected candidates

Compared with the previous method, two novelties are introduced in the transition probability. One is the probability distribution of the melody F0 difference between frames for evaluating the trajectory smoothness. Learned from the ADC04 training database, the distribution is approximated by the Laplace distribution (see Fig. 2). The trajectory smoothness is then modeled by

$$F(c_n, c_m) = \frac{1}{2b} \exp\left(-\frac{|f_{c_n} - f_{c_m}|}{b \cdot f_{c_m}}\right), b = 0.0077889 \quad (7)$$

where c_n, c_m represent the two candidates with frequencies f_{c_n}, f_{c_m} . Notice that c_n, c_m may be located at different analysis frames and the distance allowed for connection is three frames.

The other novelty is the integration of the spectral envelope similarity in the transition probability. This is intended to favor candidate connection with similar timbre such that the decoded stream is locked to the same source even when it becomes less dominant (smaller score).

$$A(c_n, c_m) = 1 - \frac{\sum_{h=0}^H |X_L(t_n, hf_{c_n}) - X_L(t_m, hf_{c_m})|^2}{\sum_{h=0}^H X_L(t_m, hf_{c_m})^2} \quad (8)$$

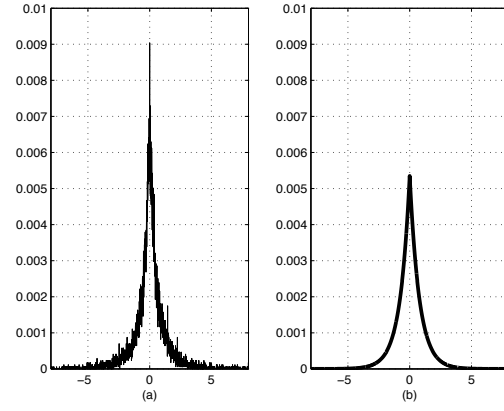


Figure 2: (a) The probability distribution of frequency deviation from ADC04 database (b) The probability density function modeled by the Laplace distribution. The x-axis is the frequency deviation in percentage.

where t_n, t_m denotes the frames where c_n, c_m are extracted. The transition probability is thus given by

$$T(c_n, c_m) = F(c_n, c_m) A(c_n, c_m)^\gamma \quad (9)$$

where γ is a compression parameter which should reflect the importance of the envelope similarity measure. In order to obtain the optimal trade-off between the emission probability (score) and the transition probability, we further apply a compression factor β on the emission probability.

The connection weight between two nodes is defined by the product of the emission probability and the transition probability, from which the forward propagated weights can be accumulated. The optimal path (melody stream) is then decoded by backward tracking through the nodes of locally maximal weights.

4. EVALUATION

In this section, we present the evaluation of the effectiveness of the perceptual criteria. Firstly, the different peak selection methods are evaluated. Then, the system with/without perceptual criteria is evaluated. Finally, the performance is compared with that of MIREX participants. The databases used are listed below:

- ADC04: 20 excerpts of about 20s including MIDI, Jazz, Pop and Opera music as well as audio pieces with a synthesized voice. It is used for our training database [28].
- MIREX05: 25 excerpts of 10-40s from the following genres: Rock, R&B, Pop, Jazz, Solo classical piano [29]. Only 13 excerpts are made publicly available.
- RWC: 100 excerpts, 80 from Japanese hit charts in the 1990s and 20 from American hit charts in the 1980s [30]. This large database is rarely used in existing publications on melody estimation.

Peak selection

To evaluate the performance of different peak selection methods, we use two metrics: *recall rate* and *mean rank*. Recall rate is the

percentage of the correct melody F0 being extracted in the candidate set. A good peak selection method shall not exclude too many peaks that support the correct F0. Mean rank is the average score ranking of the correct melody F0 in the candidate set. As long as the dominant partials of the correct F0 are selected, the resulting score shall be high and the ranking of the correct F0 be on top. For the methods implying thresholds, several values are tested in search of the best configuration. The result is shown in Fig. 3. A good configuration shall result in a point located more to the top-right corner in the figure. The reasonable results obtained seem to locate in the region of which recall rate varies from 0.85 to 0.9 and mean rank varies from 2 to 1. In general, the perceptual criteria seem to be more effective than the spectral envelopes in favoring the correct F0s.

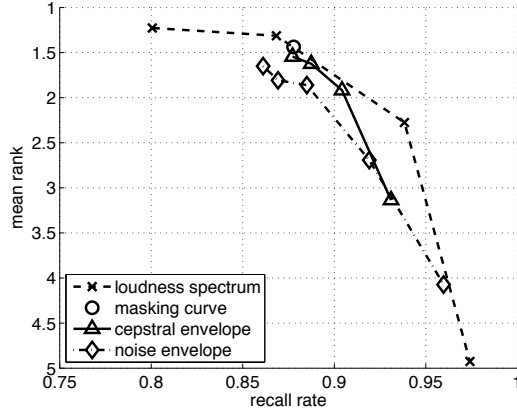


Figure 3: Evaluation results of different peak selection methods. The parameters tested are $\delta_L:(48,36,24,12)$, $\delta_C:(18,12,6,0)$ and $\delta_N:(12,9,6,3,0)$. The masking curve method does not involve any parameter and is shown as a single point.

System configurations

To understand the contribution of each component in the system, we propose to evaluate the system with different configurations. Since our current system does not detect if the melody is present (voiced) or not (unvoiced), we choose the following evaluation metrics [4]

$$\text{Raw Pitch Accuracy} = \frac{\text{number of correct estimates}}{\text{number of ground truth}} \quad (10)$$

which is defined as the proportion of the voiced frames in which the estimated F0 is within one semitone of the ground truth.

The *baseline configuration* does not take into account any perceptual properties. The peak selection simply picks the first 20 largest peaks and the tracking does not use the envelope similarity measure ($\gamma = 0$). The *perceptual configuration* uses the loudness spectrum for peak selection, the envelope similarity compression factor $\gamma = 2.4$ and the emission probability compression factor $\beta = 0.1$. These parameters are trained from the data set ADC04. For each configuration, we further evaluate how the tracking mechanism improves the average raw pitch accuracy. The results without tracking simply reports the best candidate at each frame. The

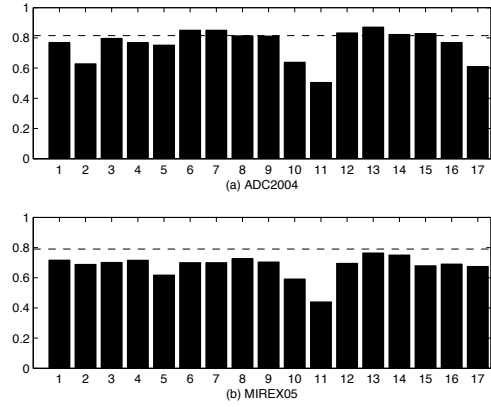


Figure 4: Raw pitch accuracy comparisons: (a) The MIREX participant results for ADC04 database (b) The MIREX participant results for MIREX05 database. The indices corresponding to MIREX participant IDs are: the first five for MIREX 2010 (HJ1, TOOS1, JY2, JY1, SG1) and the remaining twelve for MIREX 2009 (CL1, CL2, DR1, DR2, HJC1, HJC2, JY, KD, MW, PC, RR, TOOS). Please refer to MIREX website for the respective systems [16]. The horizontal line shows the results of the proposed system.

comparison is shown in Table 1. It is found that the perceptual configuration performs better than the baseline configuration by about 3 to 4%. The tracking mechanism slightly improve about 1 to 2%. Further investigation is ongoing to improve the tracking algorithm.

	best candidate	candidates + tracking
Baseline config.	73.16%	74.03%
Perceptual config.	76.92%	78.10%

Table 1: Average raw pitch accuracy for baseline configuration(without perceptual properties) and perceptual configuration. For each configuration, the frame-based estimation (reporting the best candidate) is evaluated against the tracking system.

Comparison with the state-of-the-art system

Thanks to the MIREX campaign, the performance of the start-of-the-art systems are publicly evaluated (see Fig. 4). Although the MIREX database is only partially available for our evaluation, the results (see Table 2) still demonstrate its competitive performance among the top-ranked systems.

ADC04	MIREX05	RWC
81.53%	79.00%	74.49%

Table 2: Average raw pitch accuracy of proposed system evaluated on three databases.

5. CONCLUSION

The effectiveness of perceptual properties in the context of melody estimation has been studied. For the proposed melody estimation system, the accuracy is improved by more than 3% while taking into account perceptual properties. The use of either loudness or masking curve demonstrates advantages over the proposed spectral envelope features. The envelope similarity is found to slightly improve the accuracy, too. The proposed system is evaluated on more than one hundred excerpts of music recordings and demonstrates its competitive performance to the state-of-the-art systems. Future work will be the improvement of the tracking algorithm and the development of the voicing detection algorithm.

6. REFERENCES

- [1] A. Klapuri and M. Davy, Eds., *Signal Processing Methods for Music Transcription*, Springer, New York, 2006.
- [2] M. Goto, "A real-time music-scene-description system: predominant-f0 estimation for detecting melody and bass lines in real-world audio signals," *Speech Communication (ISCA Journal)*, vol. 43, no. 4, 2004.
- [3] R. P. Paiva, T. Mendes, and A. Cardoso, "Melody detection in polyphonic musical signals: exploiting perceptual rules, note salience, and melodic smoothness," *Computer Music Journal*, vol. 30, no. 4, pp. 80–98, 2006.
- [4] G. E. Poliner, D. P. W. Ellis, A. F. Ehmann, E. Gómez, S. Streich, and B. Ong, "Melody transcription from music audio: approaches and evaluation," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1247–1256, 2007.
- [5] A. S. Bregman, *Auditory Scene Analysis*, The MIT Press, Cambridge, Massachusetts, 1990.
- [6] M. Marolt, "Audio melody extraction based on timbral similarity of melodic fragments," in *Proc. of Eurocon 2005*, 2005.
- [7] J. Salamon and E. Gómez, "Melody extraction from polyphonic music audio," Music Information Retrieval Evaluation eXchange (MIREX) 2010.
- [8] M. Marolt, "On finding melodic lines in audio recordings," in *Proc. of the Intl. Conf. on Digital Audio Effects (DAFx-04)*, 2004, pp. 217–221.
- [9] K. Dressler, "Audio melody extraction for MIREX 2009," in *5th Music Information Retrieval Evaluation eXchange (MIREX'09)*, 2009.
- [10] J.-L. Durrieu, G. Richard, B. David, and C. Févotte, "Source/filter model for unsupervised main melody extraction from polyphonic audio signals," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 564–575, 2010.
- [11] M. Ryynänen and A. Klapuri, "Transcription of the singing melody in polyphonic music," in *Proc. of the 7th Intl. Conf. on Music Information Retrieval (ISMIR'06)*, 2006.
- [12] Y. Li and D. L. Wang, "Separation of singing voice from music accompaniment for monaural recordings," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1475–1487, 2007.
- [13] K. Dressler, "Extraction of the melody pitch contour from polyphonic audio," in *1st Music Information Retrieval Evaluation eXchange (MIREX'05)*, 2005.
- [14] K. Dressler, "An auditory streaming approach on melody extraction," in *2nd Music Information Retrieval Evaluation eXchange (MIREX'06)*, 2006.
- [15] K. Dressler, "Audio melody extraction - late breaking at ISMIR 2010," in *11th Intl. Conf. on Music Information Retrieval (ISMIR'10)*, 2010.
- [16] "Music Information Retrieval Evaluation eXchange (MIREX) homepage," <http://www.music-ir.org/mirex/wiki/>.
- [17] W.-C. Chang, W.-Y. Su, C. Yeh, A. Roebel, and X. Rodet, "Multiple-f0 tracking based on a high-order HMM model," in *Proc. of the 11th Intl. Conf. on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008.
- [18] C. Yeh, *Multiple fundamental frequency estimation of polyphonic recordings*, Ph.D. thesis, Université Paris 6, 2008.
- [19] B. Bauer and E. Torick, "Researches in loudness measurement," *IEEE Trans. on Audio and Electroacoustics*, vol. 14, no. 3, pp. 141 – 151, 1966.
- [20] H. Fletcher and W. A. Munson, "Loudness, its definition, measurement and calculation," *Journal of the Acoustic Society of America*, vol. 5, pp. 82–108, 1933.
- [21] J. D. Johnston, "Transform coding of audio signals using perceptual noise criteria," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 314–323, 1988.
- [22] E. Zwicker, "Subdivision of the audible frequency range into critical bands," *Journal of the Acoustic Society of America*, vol. 33, no. 2, pp. 248–248, 1933.
- [23] ISO/IEC 13818-3, "Information technology – generic coding of moving pictures and associated audio information – part 3: Audio," Tech. Rep., ISO/IEC JTC1/SC29 WG11, 1998.
- [24] D. Schwarz and X. Rodet, *Analysis, Synthesis, and Perception of Musical Sounds*, chapter Spectral envelopes and additive + residual analysis/synthesis, pp. 175–227, Springer Science+Business Media, LLC, NY, USA, 2007.
- [25] C. Yeh and A. Roebel, "Multipl-f0 estimation for MIREX 2010," Music Information Retrieval Evaluation eXchange (MIREX) 2010.
- [26] C. M. Harris, "Pitch extraction by computer processing of high-resolution Fourier analysis data," *Journal of the Acoustical Society of America*, vol. 35, pp. 339–343, March 1963.
- [27] A. Klapuri, *Signal Processing Methods For the Automatic Transcription of Music*, Ph.D. thesis, Tampere University of Technology, 2004.
- [28] P. Cano, E. Gómez, F. Gouyon, P. Herrera, M. Koppenberger, B. Ong, X. Serra, S. Streich, and N. Wack, "ISMIR 2004 audio description contest," Tech. Rep., UPF MTG, 2004.
- [29] G. Poliner and D. Ellis, "A classification approach to melody transcription," in *11th Intl. Conf. on Music Information Retrieval (ISMIR'05)*, 2005.
- [30] M. Goto, "AIST annotation for the RWC Music Database," in *Proc. of the 7th Intl. Conf. on Music Information Retrieval (ISMIR'06)*, 2006, pp. 359–360.

Keynote 2 - Recent developments in signal processing editing and visualization - *David Zicarelli*

Over the past several decades, interactive domain-specific languages that permit the development of signal processing algorithms visually have become commonplace. I describe the advantages and disadvantages of working on algorithms in a visual way, considering factors such as cognition, performance, and reusability. Finally, I will present our recent work in visual editing of generalized synchronous DSP graphs, domain-specific graphs, visual performance monitoring, and filter design.

David Zicarelli is the founder of Cycling 74, developers of Max/MSP/Jitter, and has been working on the Max environment since 1988. Over the past 25 years he has done audio software development and research at CCRMA, IRCAM, Opcode Systems, Intelligent Music, AT&T, and Gibson Guitar.

BLACK BOX METHODOLOGY FOR THE CHARACTERIZATION OF SAMPLE RATE CONVERSION SYSTEMS

Stéphan Tassart

ST-Ericsson, STS/AEP/CP

Paris, France

stephan.tassart@stericsson.com

ABSTRACT

Digital systems dedicated to audio and speech processing usually require sample rate conversion units in order to adapt the sample rate from different signal flows: for instance 8 and 16 kHz for speech, 32 kHz for the broadcast rate, 44.1 kHz for CDs and 48 kHz for studio work. The designer chooses the sample rate conversion (SRC) technology based on objective criteria, such as figures of complexity, development or integration cycle and of course performance characterization. For linear time-invariant (LTI) systems, the transfer function contains most information necessary for the system characterization. However, being not LTI, the SRC characterization also requires aliasing characterization. When the system under study is available only through input excitations and output observations (i.e. in black box conditions), aliasing characterization obtained for instance through distortion measurements is difficult to evaluate properly. Furthermore, aliasing measurements can be messed up with weakly nonlinear artifacts, such as those due to internal rounding errors. Consider now the fractional SRC system as a linear periodically time-varying (LPTV) system whose characteristics describe simultaneously the aliasing and the in-band (so-called *linear*) behaviour from the SRC. An interesting and new compound system made of multiple instances of the same SRC system builds a LTI system. The linear features from this compound system fully characterizes the SRC (i.e. its linear and aliasing rejection behaviour) whereas weakly nonlinear features obtained from distortion measurements are only due to internal rounding errors. The SRC system can be analyzed in a black box condition, either in batch processing or real-time processing. Examples illustrate the capability of the method to fully recover characteristics from a multistage SRC system and to separate quantization effect and rounding noise in actual SRC implementations.

1. INTRODUCTION

Evaluating a digital fractional SRC system with tools designed for the evaluation of analog-to-digital convertors (ADC) is an attractive solution. For instance, the instant power measured at the output of the system excited by a swept-sine would assess the *linear* performance of the system whereas a distortion measurement would evaluate the performance of the aliasing rejection [1].

However aliasing effects are a linear effect from a polyphase system [2]. Linear multirate (MRS) [3] and linear periodically time-varying (LPTV) [4, 5] contexts are better suited for discussing fractional SRC systems. The SRC system can be equally represented by a lowpass filter H whose polyphase components form the time varying impulse response of the system. The ideal SRC is fully characterized by the conversion ratio R/P and its lowpass filter H , in particular:

- its passband characteristics (responsible for the resampled signal coloration),
- its stopband characteristics (responsible for the attenuation of the aliased and mirrored spectral images),
- the *don't care* bandwidth (which defines the bandwidth limitations for incoming signals).

This ideal SRC model is exposed in Section 2. Finite word-length representation causes internal rounding errors and undesired deviations from the ideal model. The method presented in this paper aims at separating the impact of rounding errors from the ideal characteristics in a black box methodology condition. The developments from this paper are limited to the single input single output (SISO) case. The interest follows:

- to provide a common framework for comparing the performances of different SRC algorithms,
- to assess the performances of proprietary SRC algorithms where internals are not available,
- to assess the global performance of complicated multi-stage SRC algorithms.

Section 3 details a compound system based on multiple instances of the SRC system that is LTI. The merits of this LTI compound system for the characterization of sample rate conversion system are discussed in section 3.2. This characterization method is compared to the bispectrum method, used in [6, 7, 2, 8, 9] in order to assess the performances of LPTV systems. Finally, section 4 shows on different examples how traditional black box LTI characterization methods can be used in order to assess the performances of a SRC system.

2. IDEAL MODELS AND NOTATIONS

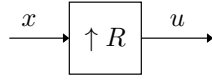
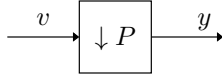
2.1. Decimation, Expansion and SRC

The building blocks for multirate systems are the delay operator, the decimation operator and the expansion operator [3]. Consider the following real (or complex) scalar signals: x , y , u and v . The expansion operator $\uparrow R$ associates u to x (see Fig. 1) in an operation sometimes referred as *upsampling*:

$$\forall n \in \mathbb{Z}, \quad u(n) = \begin{cases} x\left(\frac{n}{R}\right) & \text{if } n = kR \\ 0 & \text{else.} \end{cases} \quad (1)$$

The decimation operator $\downarrow P$ associates y to v (see Fig. 2) in an operation sometimes referred as *downsampling*:

$$\forall n \in \mathbb{Z}, \quad y(n) = v(nP). \quad (2)$$


 Figure 1: R -fold expansion operator model.

 Figure 2: P -fold decimation operator model

The previous definitions respectively for expansion and decimation can be recast in the z -transform domain. The result of the R -fold expansion (see Fig. 1 and equation (1)) verifies in the z -domain:

$$U(z) = X(z^R). \quad (3)$$

The signal y obtained as the P -fold decimation from the signal v (see Fig. 2 and equation (2)) verifies in the z -domain, with ω_P being a P^{th} root of unity:

$$Y(z^P) = \frac{1}{P} \sum_{m=0}^{P-1} V(\omega_P^m z). \quad (4)$$

The multirate system shown in Fig. 3 can be used as an ideal model for a sample rate convertor with a fractional ratio $\times R/P$ as in [5], with R and P chosen coprime. This multirate system consists of a R -fold expansion, a LTI system H referred to as the kernel of the multirate system and a P -fold decimation. The signal v is the result of the filtering of u , i.e. $v(z) = H(z) \cdot u(z)$. Thus, from Eq. (3) and (4), one obtains the modulation equation for the ideal $\times R/P$ sample rate conversion system:

$$Y(z^P) = \frac{1}{P} \sum_{m=0}^{P-1} H(\omega_P^m z) \cdot X(\omega_P^{Rm} z^R). \quad (5)$$

2.2. LPTV systems

A discrete-time linear system is defined as (L_2, L_1) -LPTV if a shift of the input by L_1 samples results in a shift of the output by L_2 samples, for any input signal [2]. Given this definition, the ideal $\times R/P$ sample rate conversion system appears as a (P, R) -LPTV system. Reciprocally, any (P, R) -LPTV system with R and P chosen as two relatively prime integers verifies equation (5) and can be characterized by a kernel H .

2.3. Polyphase type-1 decomposition

The R -polyphase type-1 components¹, $x_{\frac{k}{R}}$, are associated to a signal x in such a way that interleaving those components regenerates the original signal x . In the z -domain, it results:

$$X(z) = \sum_{l=0}^{R-1} z^{-l} X_{\frac{l}{R}}(z^R). \quad (6)$$

¹Polyphase type-2, type-3 and type-4 are alternate definitions for the polyphase decomposition, cf. [2].

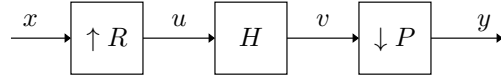
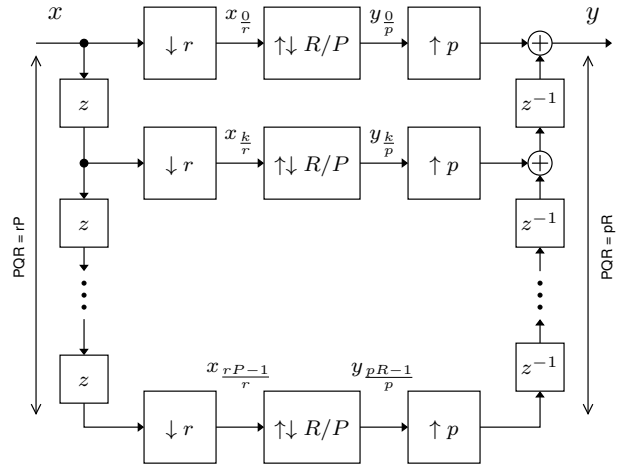

 Figure 3: $\times R/P$ sample rate conversion model.


Figure 4: LTI context for a fractional resampler.

This relationship can be inverted [3] and for any k :

$$\forall k \in \mathbb{Z}, \quad X_{\frac{k}{R}}(z^R) = \frac{z^k}{R} \sum_{n=0}^{R-1} \omega_R^{nk} X(\omega_R^n z). \quad (7)$$

3. LTI CONTEXT FOR SRC SYSTEM

3.1. Principles

Consider the compound system described on Fig. 4 where the boxes $\uparrow R/P$ represent the whole system given in Fig. 3. Consider Q a positive integer and r and p such as:

$$r = RQ, \quad p = PQ, \quad R/P = r/p.$$

The input signal x is considered as the combination of PQR interleaved channels $x_{\frac{k}{r}}$. The SRC operator $\times R/P$ is applied separately on every channel. The resulting $y_{\frac{k}{p}}$ are interleaved and form a new signal y . The left (resp. right) part of the diagram is related to the QR -polyphase analysis network (resp. PQ -polyphase synthesis network) in [7]. Note that this compound system is not causal because of the usage of the advance operator z . The following result applies:

Theorem 1. The compound system described in Fig. 4 where the ideal $\times R/P$ sample rate conversion system defined by $H(z)$, the z -transform from its kernel, is applied on PQR channels, and which input x (resp. output y) is obtained by interleaving every channel $x_{\frac{k}{r}}$ (resp. $y_{\frac{k}{p}}$), is LTI.

Furthermore, y is obtained by filtering x through the expanded kernel of the sample rate conversion system:

$$Y(z) = H(z^Q) \cdot X(z).$$

An important aspect for the proof given in the appendix is the constraint that P and R are relatively prime. In the case of a SRC system, it is always possible to chose P and R relatively prime since only the ratio R/P matters. However, the result from Theorem 1 can not apply to any (L_1, L_2) -LPTV system.

3.2. Discussion

The compound system from Fig. 4 with $Q = 1$ provides a simple methodology for assessing the performance of a SRC system. The kernel H of the SRC system is supposed to have a finite impulse response (FIR) of length L . The typical filter bandwidth is about π/R radians and the typical filter length L is about several times R (depending of the stiffness of the lowpass filter)

- Choose one test vector x from a set of possibly several test vectors.
- De-interleave the test vector x and form the PR channel test vectors $x_{\frac{k}{P}}$. Zero padding can prove to be useful in order to force the SRC system to process the signal until it has returned at rest to a steady-state.
- Apply the SRC system to each different channel vector and obtain the response channel vector $y_{\frac{k}{P}}$. The system under study being supposed to be FIR, the response channel vectors return to 0 after a maximum of $\lceil L/R \rceil$ input zero-padding samples.
- Interleave the PR response channel vectors $y_{\frac{k}{P}}$ and form the response vector y .
- Store the response vector y and repeat the process for every available test vector x .

In short, the SRC system is fed with signals in different phase situations obtained by deinterleaving a given test vector x and the output vectors are interleaved together. The analysis of the SRC system proceeds as if the test vectors x were processed by a regular LTI system. There are different subcases of interest for the test vectors.

Periodic signals are one type of interesting test vectors [7]. Once steady state is achieved (i.e after L samples on the test vector), a single output period is extracted, stored and analyzed. Note that if $N = nPR$, the channel input vectors (resp. channel response vectors) from the compound system are nP -periodic (resp. nR -periodic). Yin and Mehr in [10] use nP -periodic channel input vectors in order to excite and to identify the (R, R) -LPTV system. Transposed to the context of characterizing a SRC system, this identification method implicitly turns into a least-square FIR identification method knowing one period from x and observing one period from y . Compared to these identification methods [7, 10], the LTI approach relaxes the constraints on N which is not a necessary multiple of PR .

The impulse response is another type of interesting test vector. In such case, the impulse response $h(n)$ of the kernel H is directly available in y . When $x(n) = \delta_n$, only remains $X_{\frac{m}{R}} = z^m$ for $m \in [0, P)$; any other contribution being pure zero:

$$H(z) = Y(z) = \sum_{m=0}^{P-1} z^{-mR} Y_{\frac{m}{R}}(z^P)$$

$$\forall m \in [0, P) \quad h(nP + mR) = y_{\frac{m}{R}}(n)$$

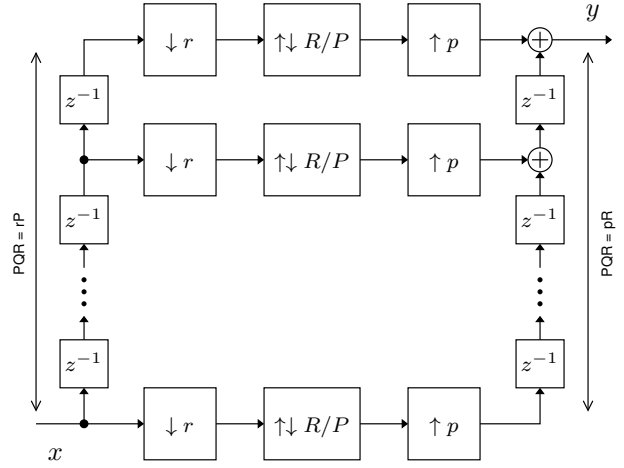


Figure 5: causal LTI context for a fractional resampler.

where n varies in the support from $y_{\frac{m}{R}}$. The method is indeed valuable and cheap (P channel vectors to process instead of PR) for characterizing an ideal SRC system but it misses the effects from internal rounding errors.

In order to cover those effects, we assume that with orthogonal input signals (obtained for instance by random phase shifting as in [7]), rounding errors average to zero. Distortion observed on the compound system is exclusively due to rounding errors: this can be observed for instance by feeding a full-scale sine in the compound LTI system. Alternatively, the method developed in [11] for measuring the performance of weakly nonlinear system can be used.

Note that the method requires the exact knowledge of the resampling ratio $\times R/P$. A separate adhoc method may be necessary in order to estimate this ratio if it is not explicitly given.

Note also that for complex resampling ratios, such as those encountered for resampling 44.1 kHz audio streams at 48 kHz, the amount of channels becomes large: for $R = 160$ and $P = 147$, we need to process $PR = 23520$ input channel test vectors. A script automating the processing and the storage of each of those PR audio files is needed.

3.3. Causal System

The LTI compound system as described in Fig. 4 is obviously not causal. In the previous discussion, non-causality was not an issue because we assumed that the analysis proceeded in batch processing. Causality may be required for real-time applications. In such case, we can use the type-4 polypase decomposition instead and obtain a causal LTI context for the analysis of a SRC in Fig. 5. The transfer function of the causal LTI compound system becomes:

$$Y(z) = z^{-PQR+1} H(z^Q) \cdot X(z).$$

3.4. Bispectrum analysis

The bispectrum (or bifrequency system function) of a LPTV system is a bivariate function $H(e^{j\omega_1}, e^{j\omega_2})$ that associates the spectrum $Y(e^{j\omega_2})$ of the input signal to the spectrum $X(e^{j\omega_1})$ of the

output signal (cf. [7]):

$$\forall \omega_2 \in \mathbb{R}, Y(e^{j\omega_2}) = \int_{-\pi}^{+\pi} H(e^{j\omega_1}, e^{j\omega_2}) X(e^{j\omega_1}) d\omega_1. \quad (8)$$

Theorem 2. The bispectrum $H(e^{j\omega_1}, e^{j\omega_2})$ of the ideal $\times R/P$ sample rate conversion system consists of Dirac lines deriving from $H(e^{j\omega})$, the transfer function of its kernel:

$$H(e^{j\omega_1}, e^{j\omega_2}) = \frac{1}{P} \sum_{m=0}^{P-1} H(e^{j\frac{\omega_1}{R}}) \delta(P\omega_1 - R(\omega_2 + 2m\pi))$$

where δ corresponds to the Dirac distribution.

Due to the fact that R and P are relatively prime, the different Dirac lines exhibited by the bispectrum reduce to one single Dirac line shaped by the transfer function of the SRC kernel and repeated along the quadrants.

Therefore the spectrum analysis of the LTI compound system compares to the bispectrum analysis of the LPTV system as in [6]. Kernel spectrum and bispectrum can both be retrieved in black box conditions. It is however our opinion that the LTI approach obtained from the compound system from Fig. 4 is more flexible than the bispectrum approach: constraints about input signal are relaxed and analysis methods are more straightforward to use because directly derived from traditional methods.

4. EXAMPLES

4.1. Upsampler $\times 3$

Figures 6 and 7 illustrate how the kernel transfer function from two $\times 3$ in-house upsampler algorithms can be revealed with the LTI methodology. The upsampling algorithms under study were provided as binary executable files that process soundfiles in double precision floating point arithmetic. The soundfiles are stored in single precision floating point (i.e. 24-bit mantissa). Both algorithms proceed in two stages, including first an upsampling block (resp. $\times 128$ and $\times 4$) and a decimating block obtained by polynomial interpolation (resp. linear interpolation and quadric interpolation) as in [12]. The actual upsampler kernel is difficult to evaluate in a formal way due to the presence of the polynomial interpolation block.

In order to reveal the upsampler kernel transfer function, two complementary test vectors, $x^{(1)}$ and $x^{(2)}$ are generated. The first test vector $x^{(1)}$ is generated as the impulse response of a low-order lowpass filter (with a cutoff frequency approximatively set to $\pi/2$). The second test vector $x^{(2)}$ was generated by the modulation of $x^{(1)}$ at the Nyquist frequency. The transfer function $H(e^{j\omega})$ is obtained as a weighted sum of the ratio $Y(e^{j\omega})/X(e^{j\omega})$ for each available test vector:

$$H(e^{j\omega}) = \frac{1}{\sum_i W^{(i)}(\omega)} \times \sum_i W^{(i)}(\omega) \frac{Y^{(i)}(e^{j\omega})}{X^{(i)}(e^{j\omega})}.$$

In this example, the weight $W^{(i)}(\omega)$ was set to $|X^{(i)}(e^{j\omega})|^2$. Fig. 6 demonstrates that in-band ripples behave accordingly to the specifications, resp. ± 0.002 dB and ± 0.01 dB. Fig. 7 demonstrates that the alias rejection performance matches the -94 dB specification only for the second algorithm. Examination of both figures demonstrates that the *don't care* bandwidth matches the specified band $[0.9\pi, 1.1\pi]/3$.

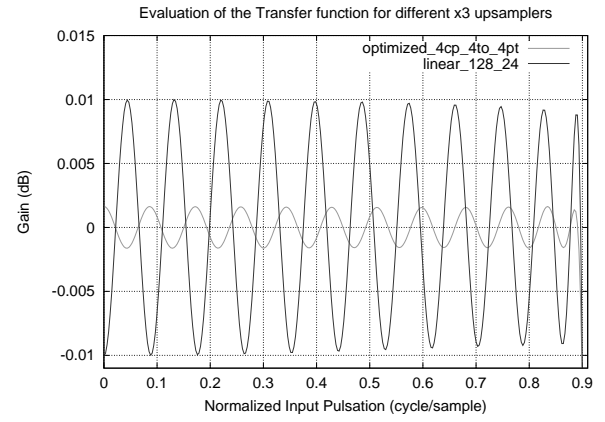


Figure 6: Estimated upsampler $\times 3$ transfer function, passband details.

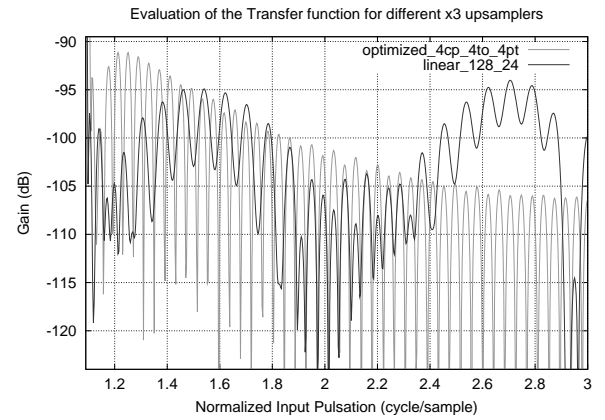


Figure 7: Estimated upsampler $\times 3$ transfer function, stopband details.

4.2. Resampler $\times 3/2$

Figures 8 and 9 illustrate how different types of arithmetic impact measurements obtained from the LTI methodology. The SRC system under study is an in-house one-stage $\times 3/2$ SRC implementation available respectively in double precision floating point arithmetic, in 24-bit fixed-point arithmetic and in 16-bit fixed-point arithmetic. The algorithms were provided in an executable format.

Fig. 8, obtained with the set of test vectors from section 4.1, illustrates the impact of the filter coefficients quantization. The 24-bit coefficient quantization provides an accuracy compatible with the stopband specification whereas the 16-bit quantization does not.

Fig. 9 demonstrates the impact of internal rounding errors on a pure sine located at $\pi/8$. The noise floor due to the storage format is about -186 dB. The harmonic distortion observed for the floating-point version of the algorithm serves as a reference assessing the impact from both the double-precision floating-point internal rounding errors and the quantization due to the file format storage. This reference is about -160 dB. The differences between the reference blue line and the dotted red (resp. dotted

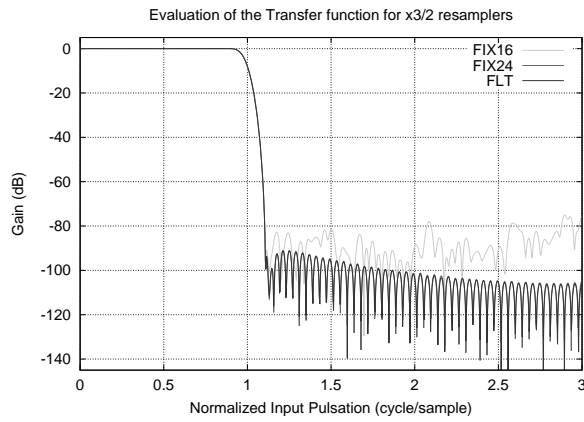


Figure 8: Transfer function of a $\times 3/2$ resampler implemented with different types of arithmetic.

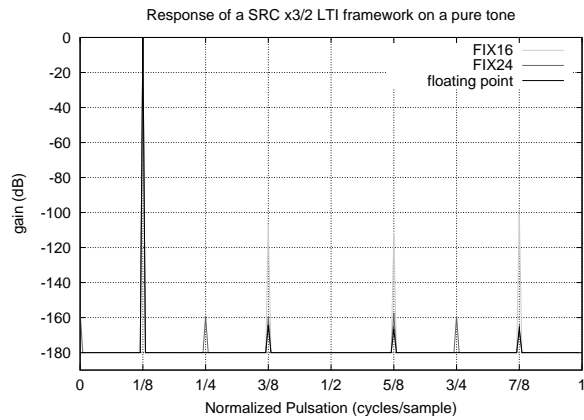


Figure 9: Impact of the internal rounding errors evaluated on a full-scale sine for a $\times 3/2$ resampler implemented with different types of arithmetic.

green) curve on Fig. 9 are only due to rounding errors in the 24-bit (resp. 16-bit) fixed-point arithmetic. In the implemented algorithm, the impact of the rounding errors in the 24-bit fixed-point arithmetic seems neglectable with regard to the alias level. In the 16-bit arithmetic, the impact of internal rounding error is more sensible but still limited with regard to the actual performance of the alias rejection. Interestingly, only odd harmonics are generated in 16-bit fixed-point arithmetic. Note that the impact of the rounding errors is quite limited in this example because the SRC processing is single stage and the rounding error happens only once per output sample without propagation when the result of the double precision accumulation is cast back into single precision. A more complex situation is expected when the SRC algorithm involves multiple stages or polynomial interpolation.

5. CONCLUSION

In this paper, we have discussed the merits of a compound system made of several instances of a sample rate conversion systems. This compound system is LTI and its transfer function, $H(z^Q)$, is

directly related to the transfer function of the SRC kernel. Regular identification and characterization methods designed for LTI systems can be applied on this system in order to reveal the linear characteristics responsible for both the in-band and the aliasing behaviour and the weakly nonlinear characteristics due to the propagation of rounding errors. This LTI context simplifies the analysis method proposed in [7] when the periods from the LPTV systems are relatively prime.

6. REFERENCES

- [1] Walt Kester, Ed., *The Data Conversion Handbook*, Elsevier, 2005.
- [2] R.L. Reng, "Polyphase and modulation descriptions of multirate systems - A systematic approach," in *Proc. Int. Conf. DSP*, Limassol, Cyprus, June 1995, pp. 212–217.
- [3] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, 1993.
- [4] R. Meyer and C. Burrus, "A unified analysis of multirate and periodically time-varying digital filters," *Circuits and Systems, IEEE Transactions on*, vol. 22, no. 3, pp. 162–168, Mar. 1975.
- [5] A.S. Mehr and T. Chen, "Representations of linear periodically time-varying and multirate systems," *Signal Processing, IEEE Transactions on*, vol. 50, no. 9, pp. 2221–2229, Sept. 2002.
- [6] R.L. Reng and H.W. Schüßler, "Measurement of aliasing distortions and quantization noise in multirate systems," in *Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International Symposium on*, May 1992, vol. 5, pp. 2328–2331.
- [7] F.A. Heinle and H.W. Schüßler, "An enhanced method for measuring the performance of multirate systems," in *In Proc. Int. Conf. on Digital Signal Processing*, Limassol, 1995, pp. 182–187.
- [8] F. Heinle, R. Reng, and G. Runze, "Symbolic analysis of multirate systems," *Maple Technical Newsletter*, vol. 3, no. 1, 1996, Special Issue featuring Engineering Applications.
- [9] F.A. Heinle and H.W. Schüßler, "Measuring the performance of implemented multirate systems," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, May 1996, vol. 5, pp. 2754–2757.
- [10] W. Yin and A.S. Mehr, "Identification of linear periodically time-varying systems using periodic sequences," in *Control Applications, (CCA) Intelligent Control, (ISIC), 2009 IEEE*, July 2009, pp. 1455–1459.
- [11] H.W. Schüßler, Y. Dong, and R. Reng, "A new method for measuring the performance of weakly nonlinear systems," in *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, May 1989, vol. 4, pp. 2089–2092.
- [12] O. Niemitä, "Polynomial interpolators for high-quality resampling of oversampled audio," <http://www.student.oulu.fi/~oniemita/dsp/deip.pdf>, Aug. 2001.

A. PROOFS FOR THE THEOREMS

Proof for Theorem 1. Let Q be any positive natural integer. Name r (resp. p) the integer $r = RQ$ (resp. $p = PQ$). The parameters P , R , r and p are associated through the following properties:

$$r = RQ, \quad p = PQ, \quad R/P = r/p, \quad P \wedge R = 1. \quad (9)$$

We use equation (7) in order to describe the signal x as the interleaving from the input channels $x_{\frac{k}{r}}$:

$$\forall k \in \mathbb{Z}, \quad X_{\frac{k}{r}}(z^r) = \frac{z^k}{r} \sum_{n=0}^{r-1} \omega_r^{nk} X(\omega_r^n z).$$

In order to continue the calculus, we introduce intermediary variables. First, let define the composite root $\Omega_{m,n}$ as:

$$\begin{aligned} \forall m \in [0, P), \quad \forall n \in [0, r), \quad \Omega_{m,n} &= \omega_p^m \omega_r^n \\ &= \omega_{PQR}^{mR+nP} \end{aligned}$$

then define the intermediary functions $\chi_l^m(z)$ and $\chi_m(z)$ as:

$$\forall l \in [0, P), \quad \chi_l^m(z) = \sum_{k=0}^{Q-1} z^{-k} X_{\frac{k+lQ}{r}}(\omega_P^{Rm} z^r),$$

$$\begin{aligned} \forall m \in [0, P), \quad \chi_m(z) &= \frac{1}{P} \sum_{l=0}^{PR-1} z^{-lQ} \chi_l^m(z) \\ &= \frac{1}{P} \sum_{l=0}^{PR-1} \sum_{k=0}^{Q-1} z^{-(k+lQ)} X_{\frac{k+lQ}{r}}(\omega_P^{Rm} z^r). \end{aligned}$$

Notice that $k + lQ$ with $k \in [0, Q)$ and $l \in [0, PR)$ covers the entire range $[0, pR)$. Therefore, the previous sum can be rewritten as:

$$\forall m \in [0, P), \quad \chi_m(z) = \frac{1}{P} \sum_{k=0}^{pR-1} z^{-k} X_{\frac{k}{r}}(\omega_P^{Rm} z^r).$$

Let evaluate $X_{\frac{k+lQ}{r}}(z^r)$ first, $X_{\frac{k+lQ}{r}}(\omega_P^{Rm} z^r)$ then

$$X_{\frac{k+lQ}{r}}(z^r) = \frac{z^{k+lQ}}{r} \sum_{n=0}^{r-1} \omega_r^{n(k+lQ)} X(\omega_r^n z).$$

For the second evaluation, notice $\omega_{rP}^{Rm} = \omega_{RP}^{Rm} = \omega_p^m$ and therefore:

$$\begin{aligned} X_{\frac{k+lQ}{r}}(\omega_P^{Rm} z^r) &= X_{\frac{k+lQ}{r}}((\omega_p^m z)^r) \\ &= \frac{(\omega_p^m z)^{k+lQ}}{r} \sum_{n=0}^{r-1} \omega_r^{n(k+lQ)} X(\omega_p^m \omega_r^n z) \\ &= \frac{z^{k+lQ}}{r} \sum_{n=0}^{r-1} \Omega_{m,n}^{k+lQ} X(\Omega_{m,n} z). \end{aligned}$$

Now, let evaluate $\chi_l^m(z)$ and finally $\chi_m(z)$:

$$\begin{aligned} \chi_l^m(z) &= \sum_{k=0}^{Q-1} z^{-k} X_{\frac{k+lQ}{r}}(\omega_P^{Rm} z^r) \\ &= \frac{z^{lQ}}{r} \sum_{k=0}^{Q-1} \sum_{n=0}^{r-1} \Omega_{m,n}^{k+lQ} X(\Omega_{m,n} z), \end{aligned}$$

$$\begin{aligned} \chi_m(z) &= \frac{1}{P} \sum_{l=0}^{PR-1} z^{-lQ} \chi_l^m(z) \\ &= \frac{1}{rP} \sum_{l=0}^{PR-1} \sum_{k=0}^{Q-1} \sum_{n=0}^{r-1} \Omega_{m,n}^{k+lQ} X(\Omega_{m,n} z) \\ &= \frac{1}{PQR} \sum_{k=0}^{PQR-1} \sum_{n=0}^{r-1} \Omega_{m,n}^k X(\Omega_{m,n} z). \end{aligned}$$

The response channels signal $y_{\frac{k}{p}}$ results from the application of the fractional SRC to $x_{\frac{k}{r}}$. Apply equation (5):

$$Y_{\frac{k}{p}}(z^P) = \frac{1}{P} \sum_{m=0}^{P-1} H(\omega_P^m z) \cdot X_{\frac{k}{r}}(\omega_P^{Rm} z^R).$$

The signal y is obtained by interleaving every channel $y_{\frac{k}{p}}$:

$$\begin{aligned} Y(z) &= \sum_{k=0}^{pR-1} z^{-k} Y_{\frac{k}{p}}(z^P) \\ &= \sum_{k=0}^{pR-1} z^{-k} Y_{\frac{k}{p}}(z^{PQ}) \\ &= \frac{1}{P} \sum_{k=0}^{pR-1} z^{-k} \sum_{m=0}^{P-1} H(\omega_P^m z^Q) \cdot X_{\frac{k}{r}}(\omega_P^{Rm} z^r) \\ &= \frac{1}{P} \sum_{m=0}^{P-1} H(\omega_P^m z^Q) \sum_{k=0}^{pR-1} z^{-k} X_{\frac{k}{r}}(\omega_P^{Rm} z^r) \\ &= \sum_{m=0}^{P-1} H(\omega_P^m z^Q) \chi_m(z). \end{aligned}$$

This concludes the evaluation of $Y(z)$ in term of the modulation components $X(\Omega_{m,n} z)$:

$$Y(z) = \frac{1}{PQR} \sum_{m=0}^{P-1} \sum_{k=0}^{pR-1} \sum_{n=0}^{r-1} H(\omega_P^m z^Q) \Omega_{m,n}^k X(\Omega_{m,n} z).$$

Since P and R are coprime, $mR + nP \bmod (PQR)$ covers exactly the range $[0, PQR)$ when $m \in [0, P)$ and $n \in [0, RQ)$. Let apply the Chinese remainder theorem: the double sum in m and n can be replaced by a simple sum in i , with $m = i \times \bar{R}$, labelling \bar{R} the multiplicative inverse from R in the ring $\mathbb{Z}/P\mathbb{Z}$:

$$\begin{aligned} Y(z) &= \frac{1}{PQR} \sum_{i=0}^{PQR-1} \sum_{k=0}^{pR-1} H(\omega_P^{i\bar{R}} z^Q) \omega_{PQR}^{ik} X(\omega_{PQR}^i z) \\ &= \frac{1}{PQR} \sum_{i=0}^{PQR-1} \sum_{k=0}^{pR-1} \omega_{PQR}^{ik} H(\omega_{PQR}^i z^Q) X(\omega_{PQR}^i z). \end{aligned}$$

Since ω_{PQR} is a root of the unity, we have:

$$\sum_{k=0}^{pR-1} \omega_{PQR}^{ik} = PQR \sum_n \delta_{i-nPQR}.$$

Every modulation component vanishes from $Y(z)$ and only remains:

$$Y(z) = H(z^Q) \cdot X(z). \quad (10)$$

This concludes the demonstration because the vanishing of every modulation component $X(\omega_{PQR}^i z)$ from $Y(z)$ proves that the system is LTI. \square

Proof for Theorem 2. The proof is obtained by substituting the expression of the bispectrum in equation (8). The Dirac lines, located at $\omega_1 = R/P (\omega_2 + 2m\pi)$, simplify the integral expression:

$$Y(e^{j\omega_2}) = \frac{1}{P} \sum_{m=0}^{P-1} H\left(e^{j\frac{\omega_2 + 2m\pi}{P}}\right) X\left(e^{j\frac{R}{P}(\omega_2 + 2m\pi)}\right).$$

In order to simplify the notation from the previous expression, introduce ω such as $\omega_2 = P\omega$:

$$Y(e^{jP\omega}) = \frac{1}{P} \sum_{m=0}^{P-1} H\left(e^{j(\omega + \frac{2m\pi}{P})}\right) X\left(e^{jR(\omega + \frac{2m\pi}{P})}\right).$$

The modulation equation (5) that characterizes the ideal $\times R/P$ SRC system can be recognized here, where z is substituted by $e^{j\omega}$. This concludes the proof by identification of the bifrequency function. \square

OPTIMAL FILTER PARTITIONS FOR REAL-TIME FIR FILTERING USING UNIFORMLY-PARTITIONED FFT-BASED CONVOLUTION IN THE FREQUENCY-DOMAIN

Frank Wefers, Michael Vorländer

Institute of Technical Acoustics,
RWTH Aachen University

{fwe,mvo}@akustik.rwth-aachen.de

ABSTRACT

This paper concerns highly-efficient real-time FIR filtering with low input-to-output latencies. For this type of application, partitioned frequency-domain convolution algorithms are established methods, combining efficiency and the necessity of low latencies. Frequency-domain convolution realizes linear FIR filtering by means of circular convolution. Therefore, the frequency transform's period must be allocated with input samples and filter coefficients, affecting the filter partitioning as can be found in many publications, is a transform size $K=2B$ of two times the audio streaming block length B .

In this publication we review this choice based on a generalized FFT-based fast convolution algorithm with uniform filter partitioning. The correspondence between FFT sizes, filter partitions and the resulting computational costs is examined. We present an optimization technique to determine the best FFT size. The resulting costs for stream filtering and filter transformations are discussed in detail. It is shown, that for real-time FIR filtering it is always beneficial to partition filters. Our results prove evidence that $K=2B$ is a good choice, but they also show that an optimal FFT size can achieve a significant speedup for long filters and low latencies.

Keywords: *Real-time filtering, Fast convolution, Partitioned convolution, Optimal filter partitioning*

1. INTRODUCTION

The term *fast convolution* summarizes techniques to efficiently compute the discrete convolution of two sequences $x(n)$ and $h(n)$. This paper considers real-time linear FIR filtering, where a *continuous stream* of input samples $x(n)$ is convolved with an impulse response $h(n)=h_0,\dots,h_{N-1}$ of finite length N . Thereby a continuous stream of output samples $y(n)$ is generated, which is defined by the discrete convolution formula

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) \cdot h(n-k) \quad (1)$$

Efficient real-time FIR filtering is important for many fields in technology and science. Its use for audio applications includes real-time digital audio effects, like equalizers and convolution reverbs, audio rendering for computer games and auralization in

virtual reality, comprehensive 3-D sound reproduction techniques, like wave-field synthesis (WFS)—and much more.

The filtering in eq. 1 can be easily implemented in the time-domain, using a direct-form FIR filter (transversal filter, tapped delay-line). The method has no inherent latency. Since filter coefficients can be directly altered, it allows concepts for a large variety of filter adaptation. Unfortunately, FIR filtering in the time-domain is very inefficient. The runtime-complexity for filtering N samples with N filter coefficients is within $O(N^2)$ and makes real-time filtering with long filter impulse responses virtually impossible.

Fast convolution methods

Methods that compute the discrete convolution faster than in $O(N^2)$ are known as *fast convolution algorithms*. Many of these techniques base on fast frequency-domain transforms and compute the discrete convolution within the transform's domain, where the convolution operation itself is computationally cheap to realize. The overall complexity is eventually determined by the fast frequency-domain transform algorithms, which have typical runtimes in $O(N \log N)$. The most famous transform is the Discrete Fourier Transform (DFT) implemented using Fast Fourier Transform (FFT) algorithms. But other transforms like the Discrete Trigonometric Transforms (DTTs) [1] or Number Theoretic Transforms (NTTs) [2] can be used to implement fast convolution as well, yielding to different formulations of the convolution operation in the transform's domain. A great deal of research has been spent on finding the minimum number of arithmetic operations to implement discrete convolution. For small sizes, number theoretic convolution concepts, like the Agarwal-Cooley algorithm [3], require less arithmetic operations. But for long filters FFT-based convolution is most efficient [3]. The popularity of FFT-based fast convolution algorithms is to a high degree reasoned by the availability of highly-optimized FFT libraries (e.g. FFTW [4], Intel Performance Primitives) and excellent CPU support of floating-point arithmetic, including instruction sets that ease the implementation of complex-valued operations.

When it comes to real-time filtering, simple frequency-domain convolution concepts, like the classical (unpartitioned) block convolution [5], lack efficiency. Filters need to be *partitioned* in order to combine computational efficiency and the demand for low input-to-output latencies. Splitting filters into several sub-filters of equal length is referred to as *uniformly partitioned convolution*. Highly-developed algorithms exist. We give an over-

view on the state-of-the-art for these techniques in section 4. An advantage of uniformly partitioned convolution is, that it can be easily implemented and suits the requirement of most applications. Another feature is that it allows combining frequency-domain filters—in serial or parallel [6]. However, when it comes to long filters, a non-uniform filter partitioning is more favorable [7,8,9,10]. It includes longer parts as well, which eventually lower the overall computational complexity. Unfortunately, non-uniform methods are difficult to implement and also put up additional restrictions on the exchange of filters [11]. The question for the optimal non-uniform filter partition has been raised by several authors [9,10].

2. CONTRIBUTIONS

When implementing real-time FIR filtering by partitioned convolution in the frequency-domain, one can choose the transform sizes—e.g. FFT sizes. Basically, small transform sizes K can be efficiently computed using *codelets* [4], but they have the disadvantage of resulting in many filter parts. Large size transforms on the other hand reduce the number of filter parts, but might compute less efficiently. For real-time filtering the standard choice is a transform size of $K=2B$ two times the block length B , which can be found in numerous publications—regarding uniformly [12] and non-uniformly partitioning [9,10]. This choice seems reasonable [12], but it is unclear whether it is the optimal solution in terms of the lowest computational effort.

In this paper we therefore research the influence of the transform size used for FFT-based partitioned frequency-domain convolution and optimize it for maximum computational efficiency. The following aspects are concerned:

- The correspondence between transform sizes and the filter partitioning is derived.
- A generalized uniformly-partitioned FFT-based convolution algorithm is presented, incorporating state-of-the-art techniques. Its runtime is analyzed in detail.
- Founding on this algorithm, the FFT size is optimized for maximum computational efficiency.
- The resulting computational costs for streaming filtering and also the exchange/adaptation of filters are reviewed.
- General conclusions are drawn and the consequences to other real-time filtering techniques are discussed.

3. PROBLEM DESCRIPTION

We regard the problem of real-time FIR filtering from the perspective that filter length N and input-to-output latency, given by the audio streaming block length B , are fixed constraints. B is chosen to meet the low latency requirements—typical values are small powers of two, like 128, 256 or 512 samples. The objective is to minimize the computational effort required for the filtering. Therefore, this work concerns the FFT size K as an optimization parameter. The choice of the transform size K has consequences

on the filter partitioning and the efficiency. In this section we introduce the basic relations between these variables.

In general, the convolution of two sequences with finite lengths M , N yields to a sequence with a maximum length of $M+N-1$. The circular convolution property of the DFT [13] states that

$$y(n) = DFT_{(K)}^{-1} \{ DFT_{(K)} \{ x(n) \} \cdot DFT_{(K)} \{ h(n) \} \} \quad (2)$$

for a DFT size K . In order to realize the desired linear convolution in eq. 1 by circular convolution as in eq. 2, the lengths M , N of the two sequences must satisfy the critical condition

$$K \geq M + N - 1 \quad (3)$$

meaning that the maximum length $M+N-1$ of the result $y(n)$ fits into the DFT period of K points. Otherwise the output is time-aliased and incorrect. For a maximal computational efficiency it is advised to fully exploit a DFT period of K values. Underutilization of the DFT period K is unfavorable, because it introduces unnecessary zero-padding and increases the number of filter parts. As figure 1 shows, the DFT period K can be fully utilized, by allocating B values for the input samples and by using all of the remaining $K-B+1$ values for filter coefficients. Accordingly, the whole filter of N coefficients is split into parts of L filter coefficients determined by

$$L = K - B + 1 \quad (4)$$

The number of resulting filter parts P is given by the integer

$$P = \left\lceil \frac{N}{K - B + 1} \right\rceil \quad (5)$$

Obviously, the number of filter parts decreases with increasing DFT sizes K , because more filter coefficients can be packed into a DFT period. Valid ranges for the FFT size K are determined by

$$B < K \leq N + B - 1 \quad (6)$$

K must in any case exceed the block length $K > B$, otherwise no filter coefficients are processed. For $K \geq N+B-1$ the filter remains unpartitioned. Values $K > N+B-1$ are useless, because they increase costs by processing ineffective, padded zeros.

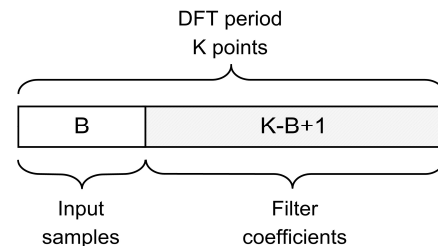


Figure 1: Effective utilization of a full DFT period for fast circular convolution without time-aliasing.

The simplest case is not to partition the filter and process it as a whole ($P=1$). Therefore, the DFT size K is chosen so that input block and filter fit into the period, meaning $K_{\text{unpart.}}=B+N-1$. The unpartitioned method is efficient for filter length $N \approx B$ close to the block length B , ideally $N=B$. For large $N \gg B$ far too much ineffective zeros are processed, making the method inefficient.

4. CONVOLUTION ALGORITHM

For our research of optimal FFT sizes we introduce a generalized partitioned convolution algorithm that allows FFT sizes to be freely adapted. It is illustrated in figure 2. The algorithm uses a uniform filter partitioning as discussed before. Input parameters are the block length B , corresponding to the desired input-to-output latency, and the filter length N . The DFT period of K points gets fully utilized with B input samples and $L=K-B+1$ filter coefficients. Accordingly, the filter of N coefficients is split into P parts of $L=K-B+1$ coefficients each. The algorithm incorporates several improvements, which have been published in recent years. The Overlap-Save scheme [13] is used to filter consecutive stream blocks. Overlap-Save computes more efficiently than Overlap-Add, because it saves extra additions of the partial outputs. Necessary delays for the subfilters, are directly implemented in the frequency-domain, using a frequency-domain delay-line (FDL) [14]. This is possible, because all DFT spectra share the same size. An FDL is implemented as a shift register of DFT spectra. Moreover, it is beneficial to implement the summation of the subfilters' results in the frequency-domain as well. Using these two techniques, only one FFT and one IFFT have to be computed for each processed stream block. Thereby the major computational load goes back to the complex-valued multiplications. Specialized FFTs/IFFTs for real-valued input data [15] are used and all computations are performed on complex-conjugate symmetric DFT spectra, speeding up the processing by nearly a factor of two.

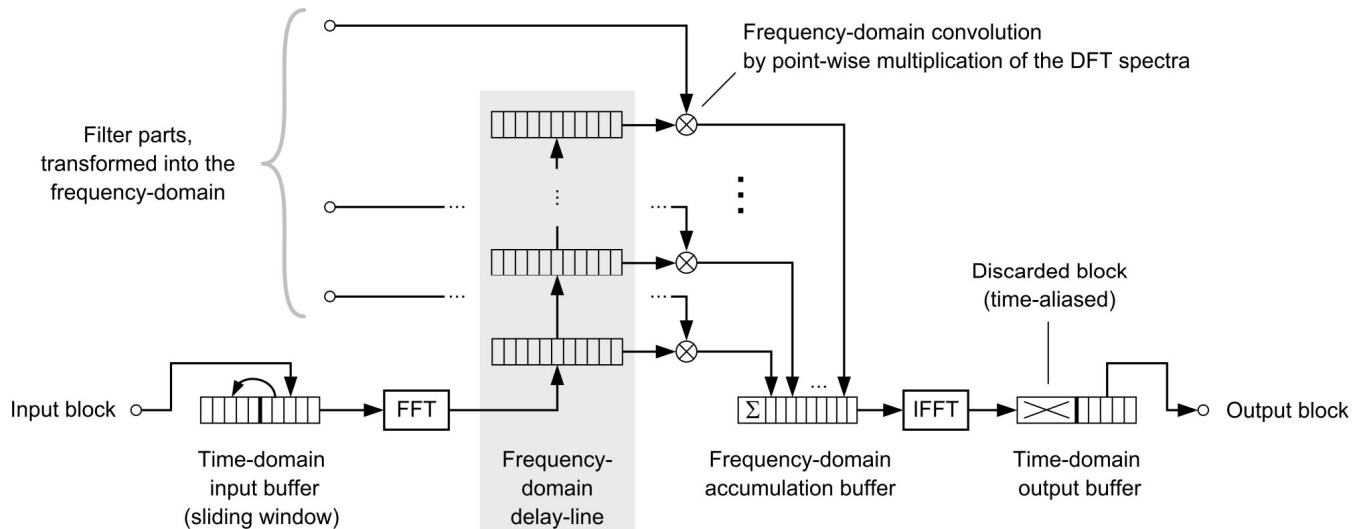
The algorithm consists of two main parts: filtering the samples of the audio streams, referred to as *stream processing*. Before they can be used with the method, a *filter transformation* has to be performed, which transforms the filter impulse responses into the according partitioned frequency-domain representation. There-

fore, it is uniformly partitioned into filter parts of the length L . Each filter part is zero-padded to match the FFT size K . Afterwards, each padded filter part is transformed using a K -point real-to-complex FFT.

Each block of the audio stream is processed in the following way: A time-domain input buffer acts as a sliding window of K samples on the stream of input samples. With each new input block, its contents are shifted $K-B$ elements to the left and the new input block of B samples is then placed to the right. The whole buffer is then transformed using a K -point real-to-complex FFT and the resulting DFT spectrum is stored in a *frequency-domain delay-line (FDL)*. Before this step, the FDL is shifted by one slot. All DFT spectra in the FDL are now point-wise complex-valued multiplied with the corresponding DFT spectra of the transformed filter parts. All results are summed up in a frequency-domain accumulation buffer. Next the contents of this buffer are transformed back into the time-domain using a K -point complex-to-real IFFT. The B left values form the output block. The other $K-B$ values are time-aliasing and discarded.

Runtime analysis

We account the computational complexities by numbers of required arithmetic operations. These measures found on theoretical considerations. Under knowledge of the properties of the given hardware, they can be approximately translated into CPU cycles or runtimes. An exact mapping however is nearly impossible to achieve, because the runtime behaviour is hard to predict—due to cache utilization and efficiency under load of multiple threads. We assume that a K -point Fast Fourier Transform (forward and backward) can be computed with $k \cdot K \log K$ arithmetic operations (with \log the natural logarithm). k is a scaling factor that depends on the actual FFT algorithm used. We benchmarked the single-threaded execution of real-valued FFTs using the FFTW3 library on an Intel Core2 system. For input sizes that are powers of two, we obtained a value of $k \approx 1.7$ —assuming one arithmetic operation per CPU cycle. This scaling factor is also a good approximation for the number of arithmetic operations of the real-valued split-radix FFT in [15]. Allowing for an effective analysis, we consider idealized costs of the FFT with a constant $k=1.7$ for arbitrary input sizes K in the following.



A complex-valued multiplication of the form $(a + bi)(c + di) = (ac - bd) + (ad + bc)i$ requires six arithmetic operations (four multiplications and three additions). DFT spectra $X(k)$ of purely real-valued input sequences $x(n)$ fulfil the Hermitian symmetry $X(k) = \overline{X(K-k)}$ [13], for a transform size of N . This symmetry can be exploited for improved performance, because only the number of

$$\left\lceil \frac{K+1}{2} \right\rceil \quad (7)$$

symmetric DFT coefficients out of the total N DFT coefficients need to be stored and processed. The complex-valued multiplication of two symmetric DFT spectra therefore takes $6\lceil(K+1)/2\rceil$ operations. The accumulation of DFT spectra is realized by point-wise additions of the elements, which accounts to $2\lceil(K+1)/2\rceil$ operations. Note that for the accumulation of P spectra only $(P-1)$ spectrum additions need to be carried out. Measures shall be independent of the block length and are therefore divided by B .

The computational cost for filtering *one output sample* is hence given by

$$T_{stream}(N, B, K) := \frac{1}{B} \left[2kK \log(K) + 6 \left\lceil \frac{N}{K-B+1} \right\rceil \left\lceil \frac{K+1}{2} \right\rceil + 2 \left(\left\lceil \frac{N}{K-B+1} \right\rceil - 1 \right) \left\lceil \frac{K+1}{2} \right\rceil \right] \quad (8)$$

The overall number of arithmetic operations for transforming a filter into the frequency-domain representation, demanding to compute P K -point FFT transforms, is expressed by

$$T_{trans}(N, B, K) := kK \log(K) \left\lceil \frac{N}{K-B+1} \right\rceil \quad (9)$$

5. OPTIMIZATION PROBLEM

For input parameters (N, B) the optimization problem is the minimization of the cost $T_{stream}(N, B, K)$ (eq. 8) for feasible transform sizes in the range $B < K \leq N + B - 1$ (eq. 6). Before discussing optimal solutions in general, we like to illustrate the characteristic properties of the cost function $T_{stream}(N, B, K)$ by the help of an example: The black curve in figure 3a shows the computational costs of the algorithm depending on the FFT size K for the example of $N=4096$ and $B=128$. For a better understanding, we also added the corresponding number of filter parts in the diagram (gray curve). For all problem instances (N, B) we found this common type of cost progression. Very small FFT sizes K just above the lower bound $K > B+1$ result in large num-

bers of filter parts, which are computationally inefficient. With increasing FFT sizes the cost decreases until the optimum K_{opt} is reached—in the example $K_{opt}=443$. From here on the costs increase again. Generally, the number of filter parts given by eq. 5 decreases with increasing FFT sizes K until it reaches the minimum of 1 for $K \geq N+B-1$ (indicated by the vertical line). From this point on the filter is unpartitioned. Larger values of K are meaningless, because unnecessary zeros are processed and the number of DFT coefficients increases. The result is a strictly linear cost progression for $K \geq N+B-1$. From the graph we can already see, that a partitioned convolution outperforms an unpartitioned filtering clearly.

Figure 3b shows the region around the optimum K_{opt} . In between the points of discontinuity, the progression is monotonously increasing and it shows ripples. These originate from ceiling in the definition of the number of symmetric DFT coefficients in eq. 7. Local minima of the cost function are located at values of K , where in eq. 7 $K+B-1$ is a factor of N —or in other words, the filter size N is a multiple of the filter part length L . At these points denoted by $K_{min}^{(i)}$ the number of filter parts is reduced by one, compared to the preceding transform size $K_{min}^{(i)}-1$. This abruptly reduces the number of necessary complex-valued

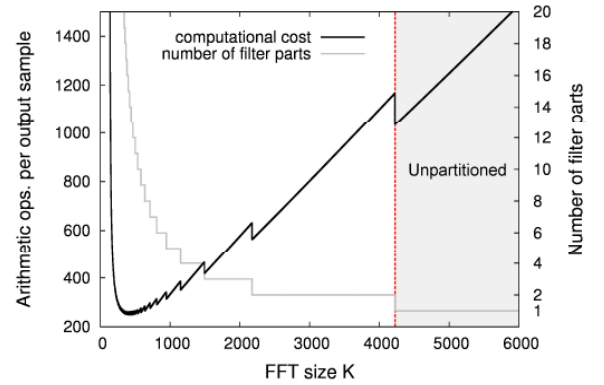


Figure 3a: Computational costs for a fixed filter length $N=4096$ and block length $B=128$ as a function of the FFT size K (black curve) The gray curve is the corresponding number of filter parts. For $K \geq 4332$ the filter consists of a single part only and remains unpartitioned.

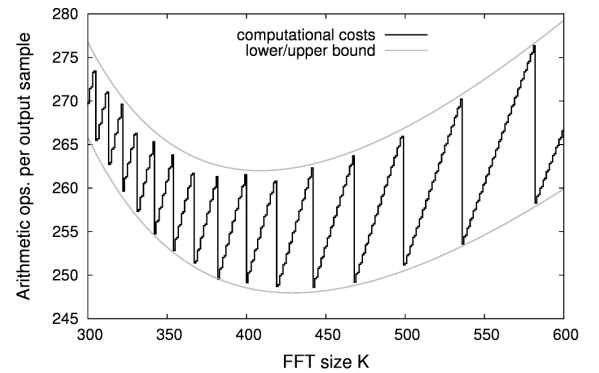


Figure 3b: Detailed view of the cost function shown in (3a) around the optimum (cost minimum), here $K_{opt}=443$.

Filter length	Block length	Optimal DFT size			Standard DFT size				Unpartitioned filter		
		K_{opt}	parts	cost	$K=2B$	parts	cost	ratio	$K_{unpart.}$	cost	ratio
1024	128	298	6	99,0	256	8	100,2	1,01	1151	242,5	2,45
1024	256	460	5	71,7	512	4	72,5	1,01	1279	136,5	1,90
1024	512	767	4	56,3	1024	2	61,2	1,09	1535	83,8	1,49
1024	1024	1365	3	47,4	2048	1	57,9	1,22	2047	57,8	1,22
4096	128	443	13	248,6	256	32	293,7	1,18	4223	1035,5	4,16
4096	256	665	10	158,9	512	16	168,9	1,06	4351	535,1	3,37
4096	512	1097	7	108,9	1024	8	109,3	1,00	4607	285,1	2,62
4096	1024	1843	5	80,2	2048	4	81,9	1,02	5119	160,2	2,00
4096	2048	3071	4	63,4	4096	2	70,6	1,11	6143	98,0	1,54
4096	4096	5461	3	53,7	8192	1	67,3	1,25	8191	67,3	1,25
16384	128	713	28	743,6	256	128	1067,7	1,44	16511	4646,3	6,25
16384	256	1075	20	431,7	512	64	554,4	1,28	16639	2342,9	5,43
16384	512	1604	15	263,7	1024	32	301,6	1,14	16895	1191,2	4,52
16384	1024	2513	11	170,9	2048	16	178,0	1,04	17407	615,4	3,60
16384	2048	4095	8	118,5	4096	8	118,6	1,00	18431	327,5	2,76
16384	4096	6826	6	88,4	8192	4	91,3	1,03	20479	183,8	2,08
16384	8192	12287	4	70,5	16384	2	80,0	1,13	24575	112,1	1,59
16384	16384	21845	3	60,0	32768	1	76,7	1,28	32767	76,7	1,28
65536	128	1257	58	2508,6	256	509	4139,5	1,65	65663	20885,9	8,33
65536	256	1745	44	1366,6	512	256	2096,4	1,53	65791	10465,0	7,66
65536	512	2559	32	768,4	1024	128	1071,1	1,39	66047	5254,6	6,84
65536	1024	4002	22	450,4	2048	64	562,3	1,25	66559	2649,4	5,88
65536	2048	6143	16	278,0	4096	32	310,7	1,12	67583	1346,8	4,85
65536	4096	9557	12	182,4	8192	16	187,3	1,03	69631	695,5	3,81
65536	8192	16383	8	128,0	16384	8	128,0	1,00	73727	370,0	2,89
65536	16384	27306	6	96,2	32768	4	100,7	1,05	81919	207,3	2,15
65536	32768	49151	4	77,6	65536	2	89,4	1,15	98303	126,3	1,63
65536	65536	87381	3	66,2	131072	1	86,1	1,30	131071	86,1	1,30

Table 1: Resulting stream filtering costs of optimal FFT sizes in comparison the other methods.

multiplications. Optimal FFT sizes K_{opt} can hence easily be found by just inspecting the absolute minimum at all points $K_{min}^{(i)}$ in the interval $[B+1, N+B-1]$.

However, it is desirable to obtain a closed formula for $K_{opt}(N, B)$ as a function of the problem instance (N, B) . This is much hindered by the discontinuous ceil functions in the cost formulation (eq. 8), which demand a piecewise analysis. The problem can be relaxed by replacing $\text{ceil}(x)$ in eq. 8 with its lower and upper bounds $x \leq \lceil x \rceil \leq x+1$ (see figure 3b). This yields to a continuous cost formulation of the functional form

$$f(x) = ax \log x + \frac{b}{x-c}$$

The absolute minimum of this continuous function is located at the zero of its derivative

$$\frac{df(x)}{dx} = a \log x + a - \frac{b}{(x-c)^2}$$

Finding the root of this type of function turned out to be difficult as well and there does not seem to be an analytic expression for x solving $df(x)/dx = 0$ (within intervals of interest), which eventually define the optimal FFT size K_{opt} .

6. RESULTS

We reviewed a multitude of problem instances (N, B) and inspected the resulting optimal transform sizes K_{opt} . In the following the results are discussed with respect to several aspects including the sheer computational cost for filtering the audio stream but also the complexity for transforming filters into the frequency-domain representation in order to use them.

Costs of stream filtering

Table 1 gives a detailed insight into the results. The two leftmost columns define the problem instance (N, B) . Followed by the data of the optimal uniformly partitioned convolution using the presented algorithm, starting with the optimal FFT size K_{opt} , the

number of resulting filter parts P and the computational costs. All cost measures in the table refer to the definition in the previous section. The next block of columns lists data for uniformly partitioned convolution with the standard FFT size $K=2B$, two times the block length B . The fourth column in this block is the cost ratio of this method in relation to the optimal approach, given by $T_{\text{stream}}(N, B, 2B)/T_{\text{stream}}(N, B, K_{\text{opt}})$. The right block lists data for the unpartitioned convolution of the instance (N, B) . Here the FFT size is chosen $K_{\text{unpart.}}=N+B-1$ and the filter consists of a single part only. The rightmost column is the cost ratio of an unpartitioned filtering in comparison to the optimal solution.

Straightaway we see that for all problem instances the two cost ratios are above one. A closer comparison on the computational costs reveal, that the optimized method is faster in any case—sometimes just by a tiny margin. The data in table 1 underlines, that an unpartitioned convolution is only efficient when $N \approx B$. Here the choice of $K=2B$ converges against the FFT size $K_{\text{unpart.}}=N+B-1 \approx 2B-1$, resulting in almost identical computational costs. We see that optimal FFT sizes K_{opt} can be smaller or larger than the FFT sizes for the two other methods. Interestingly, even for large $N \approx B$ the computational costs for an optimal FFT size K_{opt} are significantly less than for an unpartitioned filtering. Optimal filter partitions consist here of three parts and we can identify a speedup of ≈ 1.3 for the optimized method over the other methods. This is a very important discovery, stating that it is always beneficial to partition filters for real-time filtering in the frequency-domain.

Another observation is that for a filter length N , we can always find a block length B , where an FFT size of $K=2B$ results in almost optimal (minimal) costs, even if $K=2B$ differs from K_{opt} . These cases (N, B) seem to approximately fulfil $N \approx 16B$. For problem instances around this point, the standard solution $K=2B$ drops in efficiency. But we like to point out, that the penalty in costs is rather low, proving evidence that $K=2B$ is a very good choice in general. Nevertheless, when filtering very long filters with low latencies, an optimized FFT size K_{opt} leads to a significant reduction of costs. An example is the case of $N=65536$, $B=128$ where the speedup against the standard solution reaches 65%.

Costs of filter transformation

In case that filters are adapted or exchanged over time, they have first to be transformed into the corresponding frequency-domain representation. This computation introduces a separate latency, we refer to as *filter exchange latency*. It as well depends on the partitioning, as eq. 9 shows.

In figure 4 we compare filter transformation costs for an unpartitioned filter with the standard choice of $K=2B$ and the optimal transform size K_{opt} . In this example a block length of $B=128$ was chosen. The filter transformation for a transform size K_{opt} is significantly cheaper than for $K=2B$. For $N=4096$ we find a decrease in costs $\approx 22\%$ and for $N=65536 \approx 28\%$. For long filters the transformation is even cheaper than for the unpartitioned case. However, there is a lower limit in filter length where the unpartitioned filter are cheaper to realize. In the example this limit is $N \approx 4096$ taps, where $N \approx 32B$.

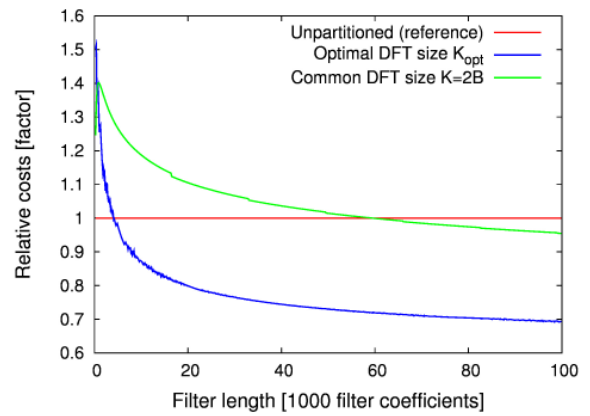


Figure 4: Relative computational costs of the filter transformation. The graph shows factors in relation to an unpartitioned fast convolution. The block length here is $B=128$.

We conclude that an optimal transform size K_{opt} lowers the computational effort for transforming filters significantly over all other methods. This is a huge advantage of our method. Not only is the stream filtering more efficient, but also the filter transformation for the majority of cases. Exceptions are found for small filter lengths. In the example we identified a maximum increase in costs of 52% for $N < 4096$. Concerning that the transformation of small filters can be computed very fast anyway, this is no real disadvantage.

7. CONCLUSIONS

In this work we discussed the choice of transform sizes for efficient real-time linear filtering realized by fast uniformly partitioned convolution in the frequency-domain. Small transform sizes result in a large number of filter parts and vice versa. For detailed research of the optimal transform size, we presented a generalized convolution algorithm with a uniform filter partitioning and analyzed its properties. Even if they found on FFT-based implementations, our results can be applied for other transform-based convolution techniques like (e.g. Discrete Trigonometric Transforms and Number Theoretic Transforms).

The presented results give a detailed insight into the properties of optimal filter partitions for uniformly partitioned frequency-domain convolution algorithms. We can confirm that the standard transform size $K=2B$ of twice the block length B —commonly found in literature—is generally a good choice, delivering a high computational efficiency. However, it turns out that specific optimization of the FFT size can significantly lower the costs for the case of long filters and low latencies. A very interesting observation is that the partitioning of filters is always beneficial and outperforms unpartitioned convolution in any case. We find for an optimal uniform partitioning, the computational costs do not have a linear dependency on the filter length—as it is known for non-uniformly partitioned convolution [9].

Consequences for other methods

The standard method for real-time filtering with long filters of $>10,000$ coefficients is non-uniformly partitioned convolution. Increasing subfilter sizes can significantly lower the computational effort compared to a uniform-partitioning. But any non-uniform filter partitioning is assembled from segments, which are basically uniformly partitioned sections with equal subfilter sizes. Consequently, our results can be applied to further optimize this class of algorithms as well and leads to an improved performance.

Applicability in practice

Optimal solutions in theory do often not translate into the desired optimal behaviour of practical implementations. A doubtful issue concerning this work might be to concern FFT sizes as an optimization parameter, without applying restrictions—for instance powers of two. A large number of FFT algorithms are known today. Efficient $O(N \log N)$ algorithms exist for arbitrary sizes (prime-factor algorithm (PFA), see [16,17]). However, FFT algorithms are most efficient if the transform size N is a highly composite number. And yet still transform sizes that are powers of two are among the most efficient. But there is no rule stating that an FFT of next greater power of two does compute faster. Therefore, it is reasonable to also account non-powers of two for implementations. Hence, our results have great importance also for practical implementations. We like to point out that the number of arithmetic operations of an FFT cannot be precisely described with a fixed scaling factor k for arbitrary input sizes. Optimal results in practice can only be achieved by benchmarking the actual runtimes on the target hardware and using these measures for the optimization.

8. REFERENCES

- [1] S. A. Martucci, "Symmetric Convolution and the Discrete Sine and Cosine Transforms," *IEEE Transactions on Signal Processing*, Vol. 42, No. 5, May 1994
- [2] J. Pollard, "The Fast Fourier Transform in a Finite Field," *Mathematics of Computation*, Vol. 25, No. 114, 1971, pp. 365-374
- [3] R.C. Agarwal, J. W. Cooley, "New Algorithms for Digital Convolution," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 25, No. 5, May 1977
- [4] Frigo, M. and S. G. Johnson, "The Design and Implementation of FFTW3," *Proceedings of the IEEE* 93 (2), 216-231 (2005).
- [5] T.G. Stockham Jr, "High-speed convolution and correlation," in *Proceedings of the April 26-28, 1966, Spring joint computer conference*. ACM, 1966.
- [6] M. Joho, G.S. Moschytz, "Connecting Partitioned Frequency-Domain Filters in Parallel or in Cascade," *IEEE Transactions on Circuits and Systems*, Vol. 47, No. 8, 2000
- [7] Gerald P. M. Egelmeers and Piet C. W. Sommen, "A new method for efficient convolution in frequency domain by nonuniform partitioning for adaptive filtering," *IEEE Transactions on Signal Processing*, vol. vol 44, 1996.
- [8] J. S. Soo, K. K. Pang, "Multidelay Block Frequency Domain Adaptive Filter," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 38, No. 2, February 1990
- [9] William G. Gardner, "Efficient convolution without input-output delay," *Journal of the Audio Engineering Society*, vol. vol 43, pp. 127-136, 1995.
- [10] Guillermo García, "Optimal filter partition for efficient convolution with short input/output delay," in *Audio Engineering Society, Convention Paper 5660*, 2002.
- [11] Christian Müller-Tomfelde, "Time-varying filter in nonuniform block convolution," in *Conference on Digital Audio Effects (DAFX-01) proceedings*, 2001.
- [12] E. Armelloni, C. Giottoli, and A. Farina, "Implementation of real-time partitioned convolution on a DSP board," *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 71-74, 2003.
- [13] Alan V. Oppenheim and Ronald W. Schaffer, "Discrete-Time Signal Processing," *Prentice Hall Signal Processing Series*. Prentice Hall, 1989.
- [14] Barry D. Kulp, "Digital equalization using fourier transform techniques," *Journal of the Audio Engineering Society*, 1988.
- [15] H. V. Sorensen, D. L. Jones, M. T. Heidman, and C. S. Burrus, "Real-valued fast fourier transform algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1987, pp. 35:849-863,
- [16] I. J. Good, "The interaction algorithm and practical Fourier analysis," *J. R. Statist. Soc. B* 20 (2), 361-372 (1958). Addendum, *ibid.* 22 (2), 373-375 (1960).
- [17] P. Duhamel and M. Vetterli, "Fast Fourier transforms: A tutorial review and a state of the art," *Signal Processing* 19, 259-299 (1990).

A SIMPLE DIGITAL MODEL OF THE DIODE-BASED RING-MODULATOR

Julian Parker

Department of Signal Processing and Acoustics,
Aalto University,
Espoo, Finland

julian.parker@aalto.fi

ABSTRACT

The analog diode-based ring modulator has a distinctive sound quality compared to standard digital ring modulation, due to the non-linear behaviour of the diodes. It would be desirable to be able to recreate this sound in a digital context, for musical uses. However, the topology of the standard circuit for a diode-based ring modulator can make the process of modelling complex and potentially computationally heavy. In this work, we examine the behaviour of the standard diode ring modulator circuit, and propose a number of simplifications that maintain the important behaviour but are simpler to analyse. From these simplified circuits, we derive a simple and efficient digital model of the diode-based ring modulator based on a small network of static non-linearities. We propose a model for the non-linearities, along with parameterisations that allow the sound and behaviour to be modified dynamically for musical uses.

1. INTRODUCTION

Ring-modulation (RM) (also known as four-quadrant modulation) is a technique similar to amplitude modulation (AM) and frequency modulation (FM) and was developed, like AM and FM, for radio transmission applications. Ideal RM is a special case of AM (i.e. the multiplication of a carrier and modulator signal), where both the carrier signal and the modulator signal are centred around 0V. When this condition is fulfilled, the carrier and modulator signals are completely cancelled and the output of the system consists of only the sum and difference frequencies of these inputs. This effect is musically useful, because it allows harmonic sounds to be transformed into clangorous inharmonic sounds whilst still retaining some of their original character and articulation.

Ring modulation was first used as musical effect by German avant-garde composers [1], notably Karlheinz Stockhausen. Ring modulators were also used by the BBC's Radiophonic Workshop in the 1950s and 1960s, notably to produce the distinctive voice of the 'Daleks' in the television show 'Doctor Who' [2]. Established as a normal part of the early electronic music studio, ring modulators were naturally included in early analog modular synthesizers. Don Buchla included a ring-modulator as one of the first modules of his System 100 synthesizer, built for Morton Subotnick [3]. Robert Moog's initial complement of modules did not include a ring-modulator, but one was later added via a collaboration with Harald Bode. Thus, the ring-modulator came to be an element available on a significant proportion of analog synthesizers, both modular and non-modular.

The name 'ring-modulation' refers to the way in which this technique was often implemented in its early days, using a configuration that employed a 'ring' of diodes [4] [5]. A schematic of this

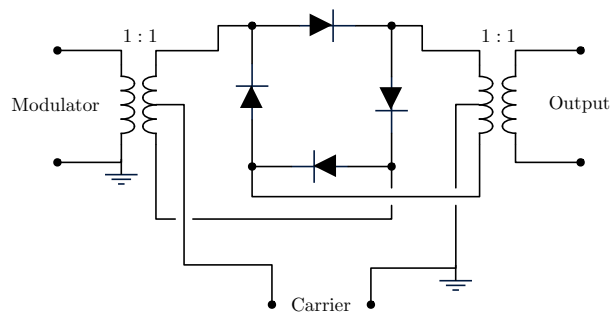


Figure 1: Schematic showing the layout of a traditional ring-modulator circuit.

type of ring-modulation circuit can be seen in Figure 1. The diodes used in the circuit have a strong effect on its final behaviour. The use of silicon diodes results in a hard clipping or 'chopping' effect and hence extremely bright extra harmonics, whilst germanium diodes produce a softer non-linearity and a 'warmer' sound [1]. Later ring modulators instead employed a structure based on VCAs, producing an output with far less added harmonics [5].

The basic idea of ring-modulation is trivial to implement digitally, as it involves only a simple multiplication of two signals. However, such a digital ring modulator lacks the additional non-linear behaviour (beyond the multiplication itself) and extra generated harmonics of a real analog ring modulator, and hence a large part of its characteristic sound. It is therefore desirable to produce a digital model that can replicate this behaviour to some extent. Previous work has derived ordinary differential equations governing the ring-modulator circuit, and solved them numerically using the Forward Euler method [6]. However, this method requires significant over-sampling (a factor of $\times 128$ is suggested), and is hence not ideal for real-time usage.

In Section 2 of this paper, we propose a simplified circuit which behaves analogously to the traditional diode ring-modulator. In Section 3 we discuss how this circuit can be modelled digitally using a network of static non-linearities. In Section 3.1 we discuss how the shape of these non-linearities can be derived, and in Section 3.2 we discuss the results produced by this algorithm. Finally, in Section 4, we conclude.

2. A SIMPLIFIED RING MODULATOR CIRCUIT

Previous work has suggested that majority of the distortion characteristics of the diode ring-modulator can be explained by assuming that only two of the diodes in the ring conduct at any one time, the

particular pair selected by the polarity of the carrier voltage [7] [8]. It is then postulated that distortion of the signal is produced by the non-linear characteristics of the diodes, and by the finite time it takes to switch between pairs [7] [8]. However, the presence of transformers in the circuit described in Figure 1 complicates the behaviour of the circuit in a number of ways, primarily by coupling voltage between nodes of the diode ring and causing, in some circumstances, three diodes to conduct simultaneously. This behaviour can be confirmed easily by measurement or by simulation in a package such as SPICE or Qucs [9].

We propose a simplified circuit which does not contain a transformer. By examining the circuit in Figure 1, we can see that if the modulator waveform is denoted V_c , the voltage at the two input nodes of the diode ring should be $V_c + V_{in}/2$ and $V_c - V_{in}/2$. We therefore replace the input transformer and V_{in} and V_c voltage sources with two voltage sources at these nodes providing the combined voltages. We then replace the output transformer with two resistors connected to ground. Figure 2 shows a schematic of this circuit. By examining this circuit, it should be clear that the diode pair D_1 and D_3 conduct when the signal V_c is positive, and the diode pair D_2 and D_4 conduct when it is negative. The signal V_{in} is present at both opposite nodes of the diode-ring, in opposing polarities, but is of smaller amplitude and assumed to not bias the diodes. The output is taken across the two resistors R_{out} (i.e. voltage $v_2 - v_4$), both of which have a high resistance value. The resistors R_{in} have a low resistance.

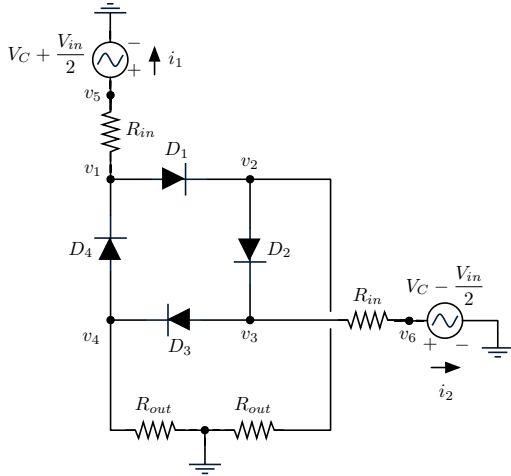


Figure 2: Schematic showing the layout of the simplified, transformer-less ring-modulator circuit.

We then make the assumption that current only flows through two diodes simultaneously, controlled by the polarity of V_c [7] [8]. We then assume that the closed diodes act as an open circuit and hence produce two substitute circuits representing the state of the circuit when V_c is positive, and when it is negative. Figure 3 shows a schematic of these circuits.

3. MODELLING THE SIMPLIFIED CIRCUIT

By examining the simplified circuit, it is easy to see what kind of digital signal processing structure could be used to replicate the

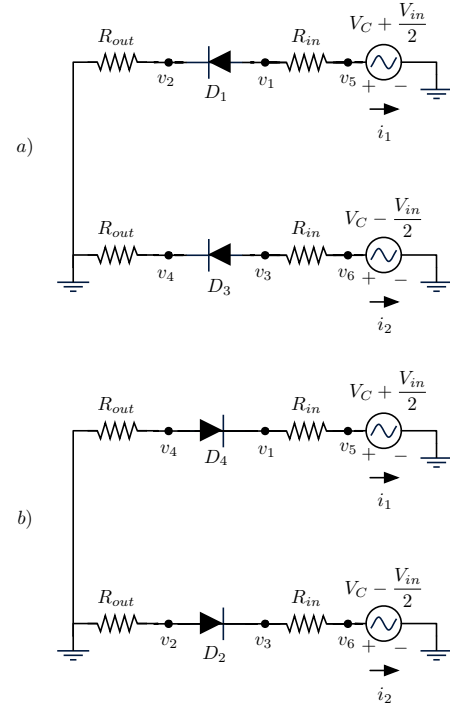


Figure 3: Schematic showing the layout of the simplified, transformer-less ring-modulator circuit when a) V_c is positive and b) V_c is negative.

circuit. There are four parallel signal paths, two which process the combination $V_c + V_{in}/2$ and two which process $V_c - V_{in}/2$. Each of these signal paths consists of a non-linearity representing the voltage-to-voltage transfer function of the resistor-diode-resistor chain, plus (as necessary) inversion to represent the direction of the diode and whether the branch of the circuit ends on v_2 or v_4 . A block diagram showing this structure is given in Figure 4. The next task is then to derive the form of the non-linearity used on each of the parallel paths of the structure.

3.1. Diode Non-Linearity Model

One possible method of modelling the diode non-linearity present in each of the branches of the simplified circuit is to use standard nodal analysis to solve for the voltage at v_2 or v_4 in one of the branches. We do this by modelling the diode using Shockley's ideal diode equation, which is given by:

$$i_D = I_S (e^{\frac{v_D}{nV_T}} - 1) \quad (1)$$

where i_D is the current through the diode, I_S is the reverse bias saturation current, v_D is the voltage across the diode, V_T is the thermal voltage and n is the ideality factor.

The exponential functions introduced by the use of Shockley's diode equation produce a relationship between the input and output voltage of the diode that is only solvable analytically by application of the Lambert W-function. Instead we expand out the exponentials using Taylor series, and then solve the resulting implicit relationship. The input-output voltage relationship of the diode when modelled in this way is given in Figure 5. The parameter values used to calculate this curve are given in Table 1, and are

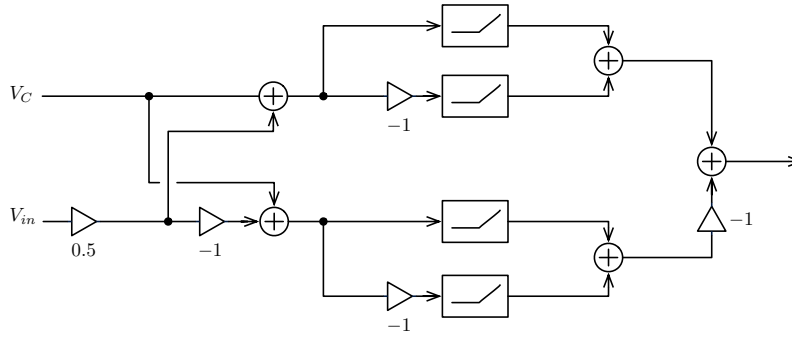


Figure 4: Block diagram showing the signal processing structure used to model the simplified ring modulator circuit.

chosen to be consistent with an average germanium diode such as the 1N34.

Table 1: Values used for the fixed parameters of the model when calculating the shaping function.

Section	Element	Value
Diodes	n	2.19
	V_T	26×10^{-3}
	I_S	10^{-12}
Fixed	R_{in}	80
Resistors	R_{out}	10^6

This method of calculating the non-linearity is only suitable for producing a look-up table, as it involves a large amount of computation. More desirable would be a non-linear function that requires few operations to calculate and which lends itself to variation by the user in order to adjust the sound of the effect. Looking at the form of the curve, it is clear that it is separated into three main sections. At low voltage, the output is approximately zero. At high voltages, the relationship is approximately affine. These sections are connected by a smooth curve. It is therefore possible to produce a very similar curve using a piecewise function, consisting of a zero section, a polynomial, and an affine section. This function is given by:

$$f(v) = \begin{cases} 0 & v \leq v_b \\ h \frac{(v-v_b)^2}{2v_L-2v_b} & v_b < v \leq v_L \\ hv - hv_L + h \frac{(v_L-v_b)^2}{2v_L-2v_b} & v > v_L \end{cases} \quad (2)$$

where v is the input voltage, v_b is a parameter specifying the equivalent of the diode forward bias voltage, v_L is a parameter giving the voltage beyond which the function is linear, and h is a parameter specifying the slope of the linear section. The shape of this function, with parameters adjusted to minimise maximum error within the region of interest compared to the modelled curve, is given in Figure 5.

3.2. Results

Figure 7 shows the output of the algorithm when driven by a 1V 500 Hz sinusoidal V_{in} and a 1V 1500 Hz sinusoidal V_C . Figure 6 shows the output of a real diode-based ring modulator, built to a design consistent with the schematic given in Figure 1 [10]. It is

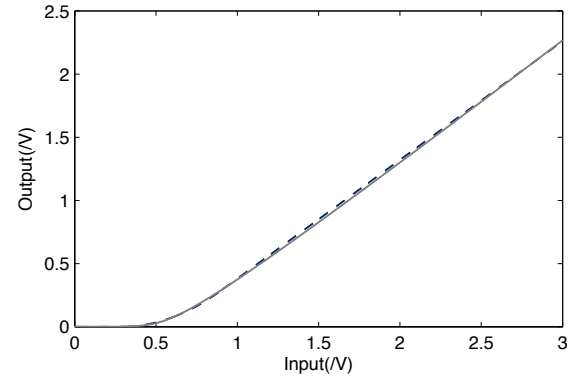


Figure 5: Graph showing the input-output relationship of the diode wave-shaper. The version derived from circuit analysis is given in solid grey, whilst the piecewise approximation is given as a dashed black line.

clear that whilst not identical, the output of the digital model is reasonably similar to the measured signal. There is a discrepancy in the peak voltage between the real and modelled ring-modulators. This is probably caused by the lack of buffering on the outputs of the real ring-modulator (as it is a passive circuit), which leads to the device having an output impedance that causes some signal loss when connected to a standard audio pre-amplifier.

Figure 8 shows the frequency spectrum of the output of model when presented with 50 Hz V_{in} and 1500 Hz V_C . As expected, visible are the sum and difference frequencies at 1550 Hz and 1450 Hz, along with further integer multiples of the modulator frequency above and below the carrier frequency. Also visible are odd harmonics of the modulated spectrum.

Informal listening tests show that the output of the algorithm is sonically satisfying when applied to synthetic material, with much of the character of a vintage ring-modulator present. When applied to more natural sound sources, for example the human voice, the result of the algorithm is harsher and brasher than that of simple digital multiplication, as would be expected. Variation of the parameters of the diode wave-shaping function, as given in Equation 2, allow a variety of distortion characters to be generated - from soft germanium-diode style saturation to the harsher distortion associated with silicon diodes. Sound examples of the algorithm applied to both synthetic and natural sources are available at the website associated with this work [11]. Evaluation of the exact accuracy of the algorithm compared to both the analog ring-modulator

and to direct numerical modelling of the circuit [6] would require further investigation. However, the author would like to emphasise that aim of the algorithm is to produce a computationally cheap vintage ring-modulator style effect, rather than to replicate the behaviour of the analog circuit exactly.

The algorithm is efficient, and could comfortably be run on modern computers in real-time with a small cost in computational load. The structure itself requires 10 operations per sample, along with 4 calls to the diode-shaping function. The diode-shaping function requires 14 operations each time it is called. In systems where it is cheap to calculate the absolute value of a sample (for example by discarding the sign of a floating point number), both the $V_C + V_{in}/2$ and $V_C - V_{in}/2$ paths can be reduced to using a single nonlinearity, by proceeding the nonlinearity with a call to an *abs()* or equivalent function.

As with any non-linear algorithm, some oversampling is recommended to avoid aliasing of the generated higher harmonics. Examination of the harmonics generated by the system reveals that they fall-off at a rate of around 20db per octave. An oversampling factor of around $\times 32$ would therefore be necessary to suppress aliasing if wide-band signals were used for both the modulator and carrier. In practical applications, the carrier is often sinusoidal, and the modulator may be a voice or instrumental signal for which the majority of the energy is in the band below 4-5kHz. In these applications, an oversampling factor of $\times 16$ or $\times 8$ is sufficient to avoid audible aliasing. The aliasing characteristics of the algorithm could be improved by re-designing the diode non-linearity function as a pure polynomial function instead of a piecewise function, and this will be an interesting topic for further work.

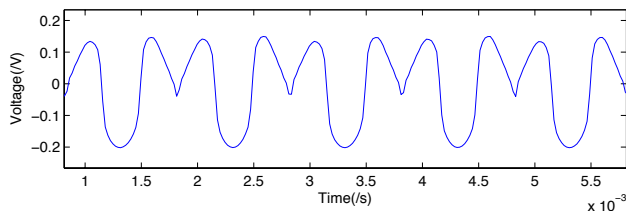


Figure 6: Output voltage produced when the real ring-modulator circuit is driven by a 500 Hz modulator and a 1500 Hz carrier at 1V

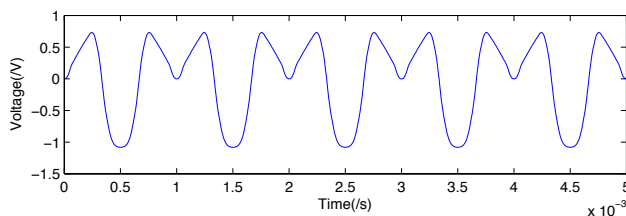


Figure 7: Output voltage produced when the model is used to process a 500 Hz modulator and a 1500 Hz carrier at 1V.

4. CONCLUSIONS

In this work we examined the operation of the diode-based ring-modulator, and proposed a simplified analogous circuit that does not contain transformers. This analogous circuit produces similar behaviour to the traditional diode ring-modulator, but its structure

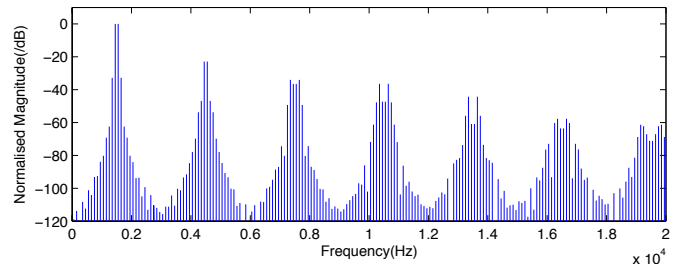


Figure 8: Spectrum of the output of the model when used to process a 50 Hz modulator and a 1500 Hz carrier at 1V.

makes it simpler to model. We then showed how this structure could be modelled digitally using a simple network of static nonlinearities. The result is an efficient parametric ring-modulator effect, which is suitable for real-time use in a computer music environment.

5. ACKNOWLEDGMENTS

This work has been financed by the Academy of Finland (project no. 122815) and by GETA. The author would like to thank Rafael Paiva for several good discussions about circuit modelling.

6. REFERENCES

- [1] H. Bode, "History of electronic sound modification," *Journal of the Audio Engineering Society*, vol. 32, no. 10, 1984.
- [2] S Marshall, "The Story Of The BBC Radiophonic Workshop," *Sound on Sound*, April 2008.
- [3] T. Pinch and F. Trocco, *Analog days: The invention and impact of the Moog synthesizer*, Harvard Univ Pr, 2004.
- [4] H. Bode, "The Multiplier Type Ring Modulator," *Electronic Music Review*, vol. 1, pp. 16–17, 1967.
- [5] Thomas E. Oberheim, "A ring modulator device for the performing musician," in *Audio Engineering Society Convention 38*, 5 1970.
- [6] R. Hoffmann-Burchardi, "Digital simulation of the diode ring modulator for musical applications," in *Proc. of the 11th Int. Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland, September 1-4, 2008.
- [7] HP Walker, "Sources of intermodulation in diode-ring mixers," *The Radio and Electronic Engineer*, vol. 46, no. 5, 1967.
- [8] RV Stewart and JG Gardiner, "Contribution to mixer intermodulation distortion of nonlinearity in the diode forward characteristics," *Electronics Letters*, vol. 7, no. 10, pp. 279–281, 1971.
- [9] ME Brinson and S. Jahn, "Qucs: A gpl software package for circuit simulation, compact device modelling and circuit macromodelling from dc to rf and beyond," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 22, no. 4, pp. 297–319, 2009.
- [10] K Stone, "Real ring modulator," <http://www.cgs.synth.net/modules/cgsrr.html> (accessed 22.10.10), 2002.
- [11] "http://www.acoustics.hut.fi/go/dafx-ringmod," .

GESTURAL AUDITORY AND VISUAL INTERACTIVE PLATFORM

B. Caramiaux*, S. Fdili Alaoui†, T. Bouchara‡, G. Parseihian§, M. Rébillat¶

LIMSI-CNRS, Ircam-CNRS, LMS-École Polytechnique
caramiau@ircam.fr

ABSTRACT

This paper introduces GAVIP, an interactive and immersive platform allowing for audio-visual virtual objects to be controlled in real-time by physical gestures and with a high degree of inter-modal coherency. The focus is particularly put on two scenarios exploring the interaction between a user and the audio, visual, and spatial synthesis of a virtual world. This platform can be seen as an extended virtual musical instrument that allows an interaction with three modalities: the audio, visual and spatial modality. Inter-modal coherency is thus of particular importance in this context. Possibilities and limitations offered by the two developed scenarios are discussed and future work presented.

1. INTRODUCTION

This paper introduces GAVIP (Gestural Auditory and Visual Interactive Platform), an interactive and immersive platform allowing for audio-graphical virtual objects to be controlled in real-time by physical gestures. GAVIP is based on a *control unit* driven by *gesture processing* monitoring the behaviour of the virtual objects. A global *control* of the virtual world ensures the *inter-modal coherency* of the proposed environment. The SMART-I² audio-graphical rendering engine achieves the *spatialization* of the virtual objects in a 3D audio-graphical scene [1]. The *spatialization* increases the coherence and thus the user's feeling to be «present» in the virtual scene [2]. *Gesture processing* allows for a wide range of natural *interaction* enhanced by the *presence* sensation [3].

GAVIP can be either considered as a sound installation, a platform of development for experimental protocol or an *extended musical virtual instrument*. By «extended», we mean that the created virtual musical instruments do not only allow an interaction with the audio modality, but an interaction with three modalities: audio, graphic and spatial. Declinations of GAVIP mainly depend on scenarios that are implemented. In this paper, focus is particularly put on two scenarios exploring the interaction between a user and the audio, graphic, and spatial synthesis of a virtual world where the interaction is thought in the sense of virtual musical instrument design. Previous works have dealt with the interaction between gesture and sound for the design of virtual musical instruments [4]. Early works have led to low interaction expressivity caused by a direct mapping between gesture parameters and sound synthesis engine parameters. Extensions have led to take into account more expressive interactions by considering higher level descriptors or

mapping based on physical models [5]. Progressively, virtual musical instrument became more linked to the sound perception and immersion. This has led to consider the spatial diffusion of sounds produced by such instruments [6], thus changing the terminology from *virtual musical instrument* to *sound installation* in which interactions are multimodal [7]. In this paper, our contribution is to propose a platform that combines sound spatialization and audio-graphical systems. To enhance interaction, our platform aims to guarantee inter-modal coherency (see Sec. 2.2).

The paper is structured as follows. In the section 2 we present the concept of GAVIP (the general platform for designing new extended musical instruments). Section 3 describes a first scenario based on a simple interaction approach. To increase the degree of inter-modal coherency, a second scenario was developed based on a physical model. It is described in Section 4.

2. THE GAVIP CONCEPT

2.1. Concept

The goal of GAVIP is the conception and the exploitation of an immersive, interactive and multimodal platform. Such a platform is built as a support for different scenarios. These scenarios will typically plunge the participants into a virtual environment populated of several audio-graphical entities among which they can freely evolve. Participants can interact in real-time with these entities through their gestures. Gesture analysis is used as a natural way to control the behaviour of audio-graphical entities and their spatialization in a 3D scene. Thus, a great effort is made to design a general architecture that allows for wide interaction strategies.

2.2. Interaction and Inter-modal coherency

In this paper, *interaction* means the bidirectional relationships between human gestures and virtual elements. Our hypothesis is that perception of interaction is enhanced by the coherency between audio and graphical objects (the so-called *inter-modal coherency*). Without being stated, this idea is shared across complementary research fields like Human Computer Interactions (HCI) [8], New Interfaces for Musical Expression (NIME) [9], or Virtual Reality (VR) [2]. By *audio-graphical object* we mean a virtual object with consistent audio and graphical components. Inter-modal coherency of the *audio* and *graphical* components of the virtual object (semantic and spatial) is required to integrate the audio and graphical streams into one unique percept [10]. However the classical approach is to synthesize separately the *audio* and *graphical* streams. Mismatches between these two information streams typically exist and are thus perceived. This can significantly degrade the quality of the audio-graphical rendering and the resulting interaction. To enhance *presence* and *interaction*, *Inter-modal coherency* has to be ensured.

* Ircam-CNRS

† LIMSI-CNRS, Ircam-CNRS

‡ LIMSI-CNRS, Univ. Paris 11

§ LIMSI-CNRS, Univ. Paris 11

¶ LIMSI-CNRS, LMS-École Polytechnique

2.3. Architecture

In order to achieve interaction with consistent audio-graphical objects in a virtual 3D scene, we designed a general virtual space where the audio and graphical parts of each entities are governed by the same control unit. Concretely, one unique control model is mapped with both modalities (the sound synthesis and the graphical synthesis) and the virtual space (spatialization) according to three consistent intra-modal mapping strategies (see Fig. 1). This allows the immersed user to influence through his/her gesture the control model itself and, consequently, the audio-graphical objects behaviour and their position in the virtual space. The consequence of an action of the user is by nature multimodal (audio/graphical/spatial) and a careful design of the mapping strategies ensures, by construction, the preservation of a strong *inter-modal coherency*.

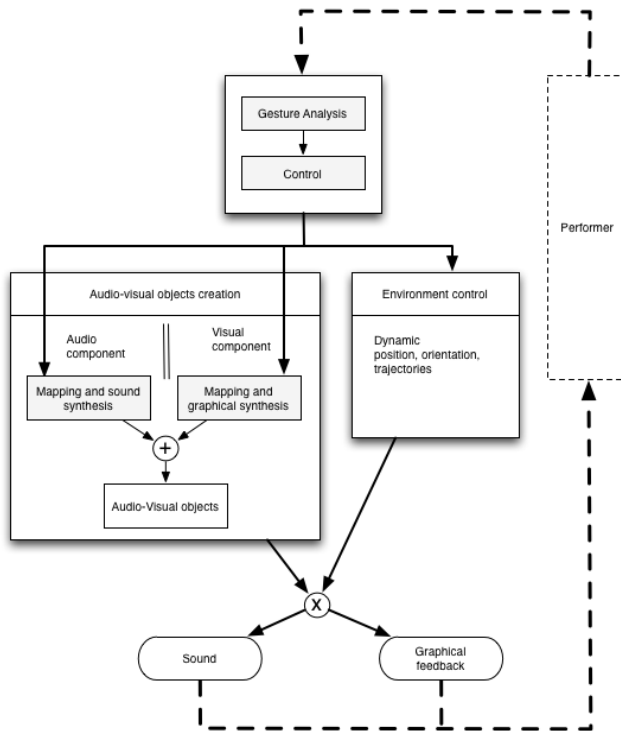


Figure 1: Overview of the general architecture of the GAVIP

2.4. Audio-graphical spatial rendering

The SMART-I² (Spatial Multi-user Audio-graphical Real-Time Interactive Interface) [1] provides an immersive, real-time and consistent audio-graphical spatial rendering. The spatial sound rendering is achieved by means of *Wave Field Synthesis* (WFS) [11]. The WFS technology physically synthesizes the sound field corresponding to one or several virtual audio sources (up to 16 sources) in a large rendering area. The 3D graphical rendering is made using passive tracked stereoscopy. In the SMART-I² system, front-projection screens and loudspeakers are integrated together to form large multi-channel loudspeakers also called *Large Multi-Actuator Panels* (LaMAPs). The rendering screens consist of two LaMAPs (2 m × 2.6 m with each supporting 12 loudspeakers) forming a corner (see Fig. 2). Overall, the SMART-I² allows to generate 16 independent virtual objects. Each virtual object is a sound source

and a graphical object. See [12] for more informations regarding audio-visual immersive systems using WFS.

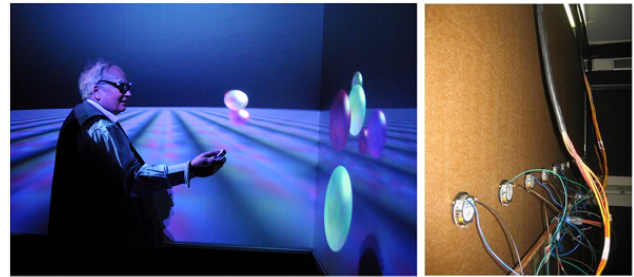


Figure 2: Front (left) and back (right) views of the SMART-I² audio-graphical rendering system. The left picture shows a participant using the early approach of interaction.

3. EARLY APPROACH

This approach was presented as a demo during the european meeting European VR-EVE 2010 that took place at the LIMSI-CNRS, Orsay, France.

3.1. Overview

In this first approach, each graphical object was a bubble and each sound source was a single sound sample taken from a given database (see Fig. 2). The challenging task was to control the set of 16 independent sound sources generated by the SMART-I². We chose to synthesize sounds of water drops. Two modes of interaction correspond to either a sequential or a single triggering of sound samples. Since a simple one-to-one mapping was impossible, we chose to have automatic controls of some of the sound parameters. We separated the *sound controls* (sample volume, sample playback speed, period between two successive sounds) from the *spatial controls* (voice number, distance, azimuth). Each of these parameters was controlled by a specific probability density function from which each parameter's value was drawn. This allowed to control the parameters' variations and to have less predictable behavior.

3.2. Gesture Control

The gesture control process is twofold. First, an Optitrack motion capture system composed of six circular cameras is used to track the user motion and adapts the graphic feedback to his/her spatial position. The calibration of the cameras is performed with the help of the Optitrack Rigid Bodies software. This software locates the markers dispatched on the user stereoscopic glasses and sends their 3D positions and orientation via a VRPN communication protocol.

Second the sound control is achieved by means of a WiiMote controller. Hence, the gyroscopes are used for driving azimuth and distances: the user can choose to be either closer or further from the sound sources and put the sources either on the right side or on the left side. Accelerometers are used for triggering control. We refer the reader to the previous section (Sec. 3.1) for the presentation of the two triggering modes. In the first mode (triggering mode), the user can use the controller to trigger percussive sound of water drop. In the second mode (sequential triggering), regular hits with the controller at a certain frequency trigger sequentially

played sounds with the same frequency. The frequency range allowed for synthesizing sound textures from separate water drops to heavy rains.

3.3. Limitations

The gesture control of audio-graphical objects proposed by this approach is straightforward. This enables a direct understanding of the general behavior by the user that can experience the scenario in a ludic way. However, this approach features several limitations. First, the global behavior is mostly controlled by probability density functions and the user's degree of control is limited. Hence the user can not really consider the interface as a virtual musical instrument as its control in sound production and positioning in space is limited. Second, the audio and graphical elements have a basic discontinuous behavior that consists in *appearing – disappearing* which does not simulate sound texture.

4. CURRENT APPROACH

To overcome the limitations of the first scenario, focus has been put on the control aspects of the platform for two main reasons. First, we aim at providing a smoother control of sound synthesis and sound spatialization. Second, in our quest for *inter-modal coherency*, we were looking for a more coherent relationship between audio and graphical parts of the virtual entities.

4.1. Overview

We refer the reader to the general architecture of GAVIP depicted in Fig. 1. The control process in the early approach was based on accelerometer–gyroscope sensor data analysis and simply linked triggering and orientation messages between gesture and audio modalities (also called *direct mapping*). Here we propose to define a more complex control unit based on a physical model simulating the interactions between elements in the scene. Audio elements are sound grains obtained by segmenting recorded environmental sounds. Since we aim to recreate sound textures we basically segment environmental textures like rains, winds, etc. Graphical sources are deformable sphere whose dynamics was inspired by magma lamp. In the following we mainly focus on audio control and rendering.

4.2. Physical Model

Here we define a physical model governing the dynamic of N abstract spherical particles in a 2D space. These particles are in a mutual interaction and in interaction with their environment. A physical model of N punctual masses linked together with springs governs their movements in this environment. Let us denote $\vec{a}_n(t)$ the acceleration of particle n at t , each element n has a mass m_n and follows the Newton's second law of motion defined by:

$$m_n \vec{a}_n(t) = \sum_{k \neq n} \vec{F}_{k \rightarrow n}(t) + \vec{F}_{\text{centre} \rightarrow n}(t) + \vec{F}_{\text{ext} \rightarrow n}(t)$$

Where at time t , $\sum_{k \neq n} \vec{F}_{k \rightarrow n}(t)$ is the sum of the forces applied from the other particles on the particle n , $\vec{F}_{\text{centre} \rightarrow n}(t)$ is the force applied from the centre on the particle n , and $\vec{F}_{\text{ext} \rightarrow n}(t)$ are the external forces exerted on n . Within the framework of this physical model, the forces are defined as follows:

- $\vec{F}_{k \rightarrow n}(t)$ accounts for: the elastic force (characterized by a parameter α_k), the viscous friction (characterized by μ_k),

the electrostatic attraction (characterized by the product of their electrostatic load $q_n q_k$)

- $\vec{F}_{\text{centre} \rightarrow n}(t)$ accounts for: the elastic attraction (characterized by K) and the central viscous friction (characterized by η)
- $\vec{F}_{\text{ext} \rightarrow n}(t)$ accounts for: the global rotation (characterized by β_n) and a magnetic field that is linked to the electrostatic load of the particle q_n .

This physical model allows to generate the relative movements of masses from a given initial state. At each time step t , the physical model returns the particles' index together with their Cartesian coordinates.

4.3. Sound Synthesis

The sound synthesis is based on the CataRT software developed by Diemo Schwarz at Ircam [13]. The software takes a set of sounds and segments them given a grain size (the default grain size is 242ms) and/or a segmentation algorithm (for instance based on onset detection). Since we work with sound textures with no onset, we choose to segment the input audio stream every 242ms. CataRT analyzes each grain by computing a set of audio descriptors. The resulting grain representation is a vector whose elements are the descriptors' mean values. CataRT then visualizes those grains in a 2D space (also called descriptor space) where the dimensions are chosen by the user among the available descriptors, for instance the *loudness* along the x -axis and the *spectral centroid* along the y -axis. Depending on an input path on this 2D space, sound grains are selected: at each time step t , the selected grain (i.e., played) is the closest to the current position x_t, y_t in the path in terms of a Mahalanobis distance. More precisely, this distance is computed between x_t, y_t and the grain descriptors mean values that correspond to the 2D space axis (e.g., loudness and spectral centroid). Note that a path in the descriptor space can be given either by using a gesture interface such as a mouse or by an input sound described in the same descriptor space. The latter case is sometimes called *mosaicing*.

Here, the sound grains are selected by the abstract particles' positions defined previously. To that end, we match both 2D representation used for the physical model and the sound corpus. Fig. 3 depicts the process. The sound corpus remains unchanged but the output of the physical model (which consists in each particle index and Cartesian coordinates) is scaled to fit the sound corpus dimensions. Hence we have two layers in the same Cartesian space (see the image at the middle of Fig. 3). An abstract particle position selects the closest grain (according to a certain neighborhood radius) at each time step. Concretely, our physical model is composed of 16 particles navigating in the sound corpus and selecting 16 grains (i.e., sound sources). Finally, this virtual plane is the sound field produced by the WFS.

4.4. Gesture control

The user's head is still tracked using an Optitrack motion capture system that allows to adapt the graphic feedback to his/her spatial position. Then, the gesture control of the virtual environment and objects corresponds to the possibilities of action and influence that a participant will have on them. One of the main challenges is to offer the participant an important power of expression in his control of the environment and the virtual objects.

At this step, the mapping between gesture analysis outputs and physical model parameters is not implemented but the global con-

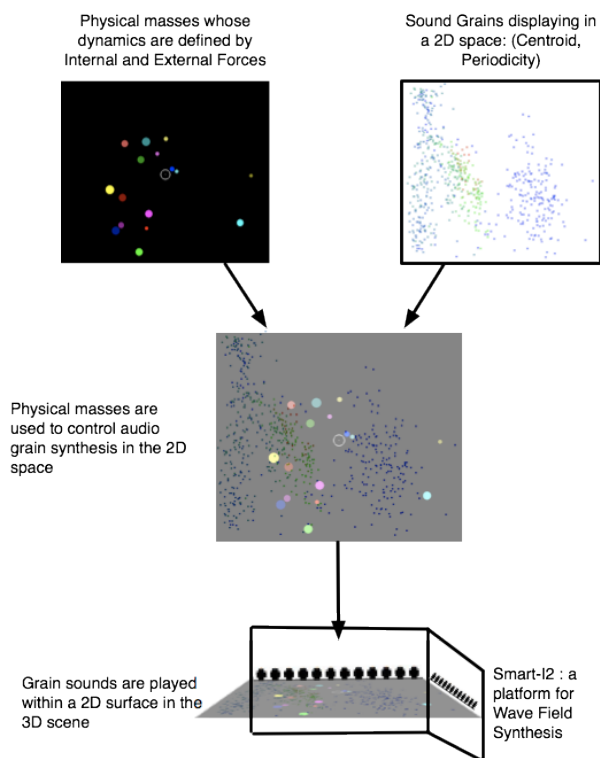


Figure 3: Sound Synthesis. Physical model environment and the sound corpus environment are matched and assigned to the sound field produced by the WFS.

cept is as follows. Two main gesture parameters will be computed (periodicity and energy) and the mapping is one-to-many. A periodic gesture with low energy will involve a stable behavior of the system with low changes in its parameters and high viscosity, and high magnetic rotation forces. A periodic gesture with high energy will decrease the viscosity and the elastic force between particles leading to a set of particles with higher quantity of motion. Finally, a non-periodic gesture with high energy will produce a highly entropic and unstable behaviour of the physical system. Hence, energy will be computed on data coming from the 3D accelerometers and periodicity on the data coming from the 3D gyroscope (using an autocorrelation criterion for example).

4.5. Implementation

A first version is already implemented in the Max/MSP real-time programming environment. A demonstration video of the physical model controlling CataRT is available online¹. A smaller version using binaural sound spatialization and simple 3D graphical rendering will be shown during the conference.

5. CONCLUSION

In this paper, a platform for the conception and exploitation of immersive, interactive and multimodal scenario were presented. An effort has been made to define a modular architecture that allows for various interesting interaction designs with respect to the coherency between audio and graphical objects. Two concrete sce-

narios were presented. The first one made use of simple triggering and orientation interactions between gesture and audio synthesis and spatialization. The second scenario proposed a general physical model that governs the behavior of audio-graphical objects in the scene. Gestures analysis controls the physical model. This new approach offered better coherency between audio and visuals as well as a smoother gesture control of them.

Future works will consist in (1) completing the implementation of the gestural control of the physical model and (2) designing perceptual experiments to evaluate the platform.

6. ACKNOWLEDGMENTS

We are grateful to both LIMSI-CNRS and UMR STMS Ircam-CNRS for the lending of equipment. We also would like to thank Matthieu Courgeon for his contribution and technical assistance on the 3D graphical programming.

7. REFERENCES

- [1] M. Rébillat, E. Corteel, and B.F. Katz, "Smart-i²: "spatial multi-user audio-visual real time interactive interface", a broadcast application context," in *Proceedings of the IEEE 3D-TV conference*, 2009.
- [2] K. Bormann, "Presence and the utility of audio spatialization," *Presence: Teleoperators & Virtual Environments*, vol. 14, no. 3, pp. 278–297, 2005.
- [3] M.J. Schuemie, P. Van Der Straaten, M. Krijn, and C.A.P.G. Van Der Mast, "Research on presence in virtual reality: A survey," *CyberPsychology & Behavior*, vol. 4, no. 2, pp. 183–201, 2001.
- [4] A. Hunt, M.M. Wanderley, and M. Paradis, "The importance of parameter mapping in electronic instrument design," in *Proceedings of the 2002 conference on New interfaces for musical expression*. National University of Singapore, 2002, pp. 1–6.
- [5] M.M. Wanderley and P. Depalle, "Gestural control of sound synthesis," *Proceedings of the IEEE*, vol. 92, no. 4, pp. 632–644, 2004.
- [6] G. Leslie, B. Zamborlin, P. Jodlowski, and N. Schnell, "Grainstick: A collaborative, interactive sound installation," in *Proceedings of the International Computer Music Conference (ICMC)*, 2010.
- [7] A. Camurri, "Multimodal interfaces for expressive sound control," in *Proceedings of the 7th International Conference on Digital Audio Effects (DAFX-04)*, Naples, Italy, 2004.
- [8] R. Vertegaal and B. Eaglestone, "Looking for sound?: selling perceptual space in hierarchically nested boxes," in *CHI 98 conference summary on Human factors in computing systems*. ACM, 1998, pp. 295–296.
- [9] X. Rodet, J.P. Lambert, R. Cahen, T. Gaudy, F. Guedy, F. Gosselin, and P. Mobuchon, "Study of haptic and visual interaction for sound and music control in the phase project," in *Proceedings of the 2005 conference on New interfaces for musical expression*. National University of Singapore, 2005, pp. 109–114.
- [10] C. Spence, "Audiovisual multisensory integration," *Acoustical science and technology*, vol. 28, no. 2, pp. 61–70, 2007.
- [11] A.J. Berkhout, D. De Vries, and P. Vogel, "Acoustic control by wave field synthesis," *Journal of Acoustical Society of America*, vol. 93, pp. 2764–2764, 1993.
- [12] D. de Vries, *Wave Field Synthesis*, Audio Engineering society Monograph, 2009.
- [13] D. Schwarz, G. Beller, B. Verbrugghe, S. Britton, et al., "Real-time corpus-based concatenative synthesis with catart," in *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, Montreal, Canada. Citeseer, 2006, pp. 279–282.

¹<http://imtr.ircam.fr/imtr/GAVIP>

TIME-VARIANT DELAY EFFECTS BASED ON RECURRENCE PLOTS

Taemin Cho[†], Jon Forsyth[†], Laewoo Kang[‡] and Juan P. Bello[†]

[†] Music and Audio Research Lab (MARL)
Music Technology, New York University
New York, USA

{tmc323, jpf211, jpbello}@nyu.edu

[‡] Department of Information Science
Cornell University
Ithaca, NY 14850 USA

lk423@cornell.edu

ABSTRACT

Recurrence plots (RPs) are two-dimensional binary matrices used to represent patterns of recurrence in time series data, and are typically used to analyze the behavior of non-linear dynamical systems. In this paper, we propose a method for the generation of time-variant delay effects in which the recurrences in an RP are used to restructure an audio buffer. We describe offline and real-time systems based on this method, and a realtime implementation for the Max/MSP environment in which the user creates an RP graphically. In addition, we discuss the use of gestural data to generate an RP, suggesting a potential extension to the system. The graphical and gestural interfaces can provide an intuitive and convenient way to control a time varying delay.

1. INTRODUCTION

Recurrence plots (RP) are binary matrices representing patterns of repetition in sequential data. They are used for analyzing and visualizing the behavior of non-linear dynamical systems, and have been applied in fields as diverse as physics, genomics, chemistry, and economics [1, 2]. More relevant, RPs have also been applied to the analysis of music, e.g. for understanding rhythmic structure [3], cover-song identification [4] and measuring structural similarity [5].

In this paper, we propose a system that inverts this process: instead of using RPs to analyze an audio sequence, our system restructures music audio using the patterns of recurrence represented by a given RP. The approach works by combining blocks of the input signal such that the repetitions characterized by the RP are enforced on the output signal. The system thus acts as a time-variant delay line, able to produce complex patterns of repetition. Furthermore, we can use a graphical or gestural interface to modify the topology of the RP, hence providing a novel mechanism for system control that is more intuitive than existing hardware and software implementations of similar effects. Finally, note that our approach operates on an audio buffer, either offline or in real time, as a digital audio effect, and is thus unlike previous approaches that synthesize audio directly from the output of non-linear systems [6].

The remainder of this paper is structured as follows. Section 2 discusses the basics of delay lines and related commercial and non-commercial work. In section 3 we briefly define recurrence plots and propose a method for adapting them to the task of restructuring audio. Section 4 describes a Max/MSP implementation of our system, while section 5 discusses a preliminary gestural control mechanism. Finally, Section 6 discusses our conclusions and some directions for future work.

2. RELATED WORK

Delay lines are widely used audio effects [7]. In their simplest form, the output signal $y[n]$ consists solely of the input signal $x[n]$ delayed by an integer number of samples M , i.e.

$$y[n] = x[n - M] \quad (1)$$

In addition, the delay can feed the output signal $y[n]$, scaled by a gain factor g , back into the input:

$$y[n] = x[n] + g \cdot y[n - M] \quad (2)$$

where $0 \leq g \leq 1$ in order to ensure stability. While in equations 1 and 2, M is restricted to integer values, fractional delay techniques allow for arbitrary delay times. This model serves as the basis for most audio delays, both in hardware and software.

Using a fixed delay time (i.e., a constant value of M) produces a regular pattern of repetition, resulting in a time-invariant system. Alternatively, some delay line implementations allow M to vary over time, either manually or by a low-frequency oscillator, thus creating irregular patterns of repetition. Techniques such as time shuffling (brassage) and granular synthesis [7], in which an output buffer consists of random combinations of segments of an input buffer, can be regarded as time varying delay effects.

Such variations are common practice in commercial hardware implementations, e.g. as pedals or rack-mounted units. A number of artists, for example guitarists David Torn and Bill Frisell, use such devices in performance to achieve various glitching, stuttering, and similar effects in real time. Frequently, these artists produce these effects by changing the delay time (and other parameters) via direct and active manipulation of various controls on the delay device. In an effort to improve these interactions, novel user interfaces have been proposed. For example, the MATRIX interface [8] is a physical controller consisting of a 12x12 array of vertical rods, each of which able to move up and down independently. By varying the pressure applied to the rods and the orientation of the hands, the user generates a continuous signal that can be used to control effects and synthesis parameters. In one example, the control data is used to continuously change the delay time parameters of a multi-tap delay.

Other time-variant delays have been implemented in environments such as Pure Data, Max/MSP, and Supercollider. For example, Jonny Greenwood, guitarist for the rock band Radiohead, uses a Max/MSP patch that repeats segments of live audio in unpredictable patterns [9]. Another example is the BBCut2 library for Supercollider [10] that allows users to splice and rearrange audio in real time or offline. BBCut2 makes use of “cut procedures,” which define how an audio buffer is subdivided, including tools

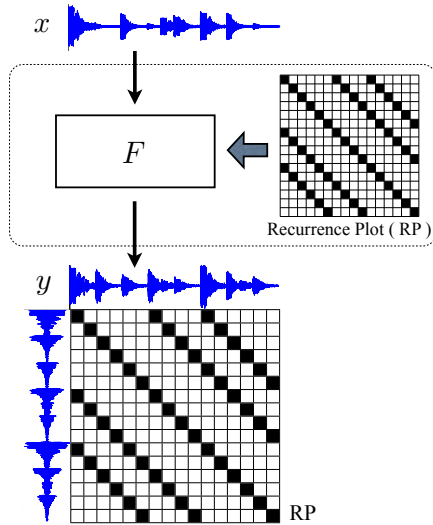


Figure 1: System overview : Input signal x is filtered by F yielding y . A given recurrence plot (RP) is a parameter set of F . As seen in the bottom half of the figure, the resulting sound y has a structural form that can be described by the given recurrence plot (RP).

to analyze the rhythmic structure of the audio. Users can create their own cut procedures programmatically, or use predefined procedures included with the library. While BBCut2 is undoubtedly a powerful tool capable of creating a wide variety of interesting effects, it is a library and not a stand-alone application with a fully developed user interface. Thus, use of BBCut2 entails a certain amount of programming knowledge, not necessarily available to most composers and performers. In addition, there are a number of freeware and commercially available plugins, such as iZotope's Stutter Edit plugin¹, that allow users to create various types of glitch and stutter edits.

3. APPROACH

While the systems and implementations discussed above can create compelling musical results in the right hands, none offer an intuitive interface that allows a user to quickly create and experiment with different time varying delay patterns. In this paper we argue that recurrence plots (RP) can be used to exercise (and visualize) control of audio delays, and propose a method for doing so, which can be seen in Figure 1. An input signal x is rearranged and remixed by a function F in order to produce the output signal y . F is fully defined by a given RP, such that the recurrences in the plot should also appear in y . In the following sections we will discuss how RPs are obtained, and describe a method for using the plots to specify F , first offline, and then in real time.

3.1. Recurrence Plots

Let us assume the time series y to describe the output of a dynamical system, such that if the state of the system at time i recurs at time j , then y_i is the same as y_j within a margin of error, ϵ . In

¹<http://www.izotope.com/products/audio/stutteredit>

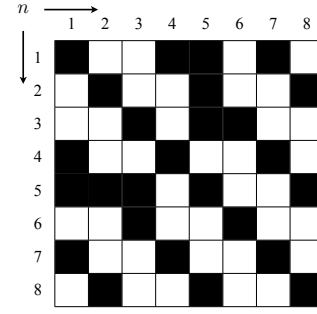


Figure 2: An example of recurrence plot with $N = 8$.

this case, cell (i, j) of the RP matrix is set to 1 (black in the graphical representation); otherwise, the cell is set to 0 (white). More formally, the RP matrix R can be defined as follows:

$$R_{i,j} = \begin{cases} 1, & \|y_i - y_j\| < \epsilon \\ 0, & \|y_i - y_j\| \geq \epsilon \end{cases}, \quad i, j = 1, \dots, N \quad (3)$$

where N represents the length of the time series, and thus the number of rows and columns in the plot, and $\|\cdot\|$ denotes a norm (e.g., Euclidean norm). The resulting plot is a symmetric, binary matrix such as the one shown in Figure 2.

3.2. Audio Restructuring using a Recurrence Plot

Next, we must develop a method for restructuring audio using an existing RP. To simplify the reconstruction problem, assume that the input signal $x[n]$ is finite and divided into N segments, $n \in [1, N]$. The output sound $y[n]$ can be obtained by a linear combination of the input segments as:

$$y[n] = \sum_{i=1}^N x[i] \cdot c_{i,n} \quad (4)$$

where $c_{i,n}$ is a reconstruction coefficient satisfying the condition that $y[n]$ should have the temporal structure described in the RP.

A simple approach to find an appropriate coefficient set c is by direct application of the information about the temporal recurrences described in the RP. That is, if a state at time n_1 recurs at time n_2 , the states at time n_1 and n_2 are assumed to be the same, i.e. $y[n_1] = y[n_2]$. As an example, consider the simple RP shown in Figure 2. In the first column of this RP, the rows at time $n = \{1, 4, 5, 7\}$ are activated, indicating that the output sounds $y[1]$, $y[4]$, $y[5]$, and $y[7]$ should be identical. In the same manner, from column 2, we can infer that $y[2] = y[5] = y[8]$, from column 3 that $y[3] = y[5] = y[6]$, etc. However, the fact that $y[5]$ appears in each of the first three columns implies that all the segments are equal (i.e. that $y[1] = y[2] = \dots = y[N]$). This result is in conflict with the given RP: if all the segments were identical, all the cells of the RP matrix would be black.

The problem with this approach is the assumption that recurring segments are identical. In fact, the margin of error ϵ from equation (3) implies that the RP represents only approximate relationships between segments. Therefore, in the previous example, while the first column indicates that $y[1] \approx y[5]$, and the second indicates that $y[2] \approx y[5]$, we cannot assume that $y[1] \approx y[2]$.

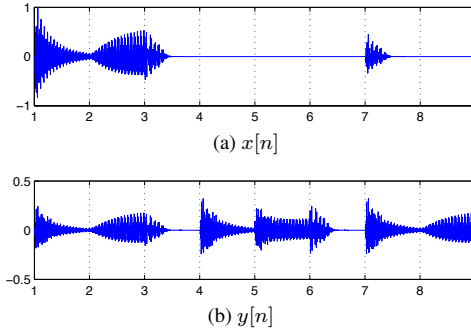


Figure 3: Reconstruction using the RP in Figure 2 : (a) the input signal $x[n]$ (b) the resulting sound $y[n]$.

Because information is lost when computing an RP (i.e., the exact distance between $y[i]$ and $y[j]$), precise reconstruction of y is impossible. However, we can approximate $y[n]$ by considering only column-wise recurrences, and ignoring any relations that appear within a given column. By doing this, each row becomes independent, indicating the recurrences of an individual component, $x[n]$. With this in mind, the coefficients $c_{i,n}$ can be derived from the n th column vector of RP as follows:

$$c_{i,n} = \frac{R_{i,n}}{\sum_{i=1}^N R_{i,n}} \quad (5)$$

where the denominator is a normalization factor accounting for the number of activated components in the column. Therefore, we can derive the following reconstruction rule from equations (4) and (5):

$$y[n] = \frac{\sum_{i=1}^N x[i] \cdot R_{i,n}}{\sum_{i=1}^N R_{i,n}}, \quad (6)$$

Figure 3 shows the input sound $x[n]$ and the resulting sound $y[n]$ using the RP in Figure 2. Accordingly, $y[1] \approx y[4] \approx y[7]$, with each output segment a combination of the input components $x[1, 4, 7]$. Likewise, $y[1] \approx y[5]$, with each output segment a combination of $x[1, 5]$.

3.3. Real-time Approach

There are two main restrictions to adapting the reconstruction approach discussed in Section 3.2 to real-time processing. First, unlike the non-real-time situation, the length of the incoming signal is unknown, and thus assumed to be infinite. Second, future audio segments are not available. The following solutions and compromises are necessary to cope with these restrictions.

First, for a given $N \times N$ plot, we use an N -length circular buffer $B[\hat{n}]$, $\hat{n} \in [1, N]$. The buffer stores the last N L -long segments of the incoming signal x , such that the oldest segment is always the one to be overwritten. For this to happen, we define the buffer index \hat{n} as follows:

$$\hat{n} = ((n-1) \bmod N) + 1, \quad n \in \mathbb{Z}^+ \quad (7)$$

such that \hat{n} is reset to 1 after each group of N signal blocks has been stored. The index \hat{n} is also used to index the rows and columns of the RP, allowing us to rewrite equation (6) as:

$$y[n] = \frac{\sum_{i=1}^N B[i] \cdot R_{i,\hat{n}}}{\sum_{i=1}^N R_{i,\hat{n}}}, \quad (8)$$

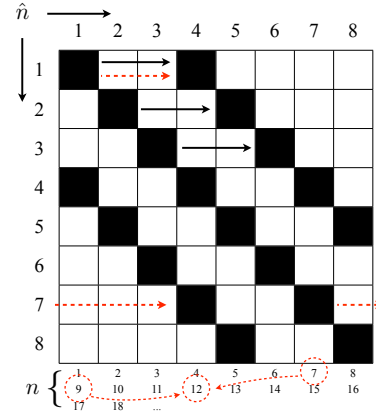


Figure 4: A given RP ($N = 8$) for the real-time process: \hat{n} indicates the corresponding row and column indexes at time n (bottom of the figure).

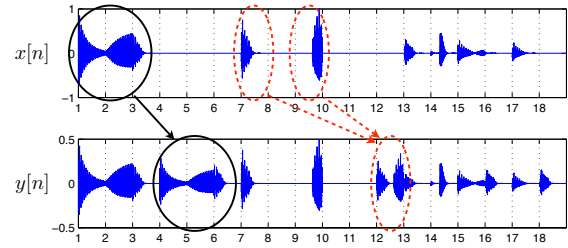


Figure 5: Real-time reconstruction example: the resulting sound $y[n]$ from the input $x[n]$. $x[1, 2, 3]$ recur at $y[4, 5, 6]$ (solid circles and arrow), and both $x[7]$ and $x[9]$ recur at $y[12]$ (dotted circles and arrows). The magnitude differences are due to normalization.

Second, the causality of the system means that the current output is necessarily a function of previous inputs. However, the use of the circular buffer introduces boundary effects that are not immediately apparent. Take for example the RP in Figure 4: $x[1]$ recurs at $n = 4$, $x[2]$ recurs at $n = 5$, and $x[3]$ at $n = 6$ (indicated by solid arrows in the figure), thus seemingly defining a standard delay with delay time 3. However, due to the modulo operation, $x[7]$ recurs at $n = 12$, a delay of 5 instead of 3. In fact, the fourth column of the RP indicates that $y[12]$ is a linear combination of $x[7, 9, 12]$ (dotted arrows and circles in the figure).

Figure 5 shows an example of real-time reconstruction using the RP in Figure 4. As described above, $x[1, 2, 3]$ recur at $y[4, 5, 6]$ (indicated by the solid arrow and circles in the figure), and both $x[7]$ and $x[9]$ are mixed into $y[12]$ (indicated by the dotted arrows and circles in the figure).

In practice, due to the fact that the buffer B is initially empty, the direct implementation of equation (8) yields an unwanted fade-in effect at the beginning of $y[n]$, for the length of one cycle of buffering (i.e., for $n \in [1, N]$). We thus modify the normalization factor in equation (8) as follows:

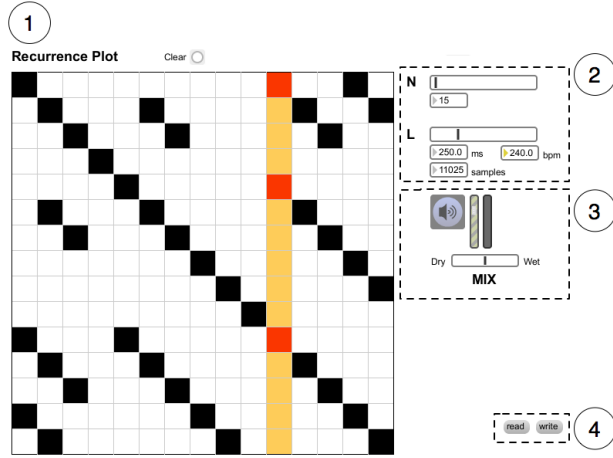


Figure 6: Max/MSP user interface.

$$y[n] = \frac{\sum_{i=1}^N B[i] \cdot R_{i,\hat{n}}}{K} \quad (9)$$

$$\text{where } K = \begin{cases} \sum_{i=1}^n R_{i,n} & n \leq N \\ \sum_{i=1}^N R_{i,\hat{n}} & n > N \end{cases}$$

4. IMPLEMENTATION

4.1. Technical details

The RP-based delay approach described above was implemented using Max/MSP, a graphical programming language designed specifically for use in realtime systems. It also provides a number of useful user interface elements, making it an excellent prototyping environment. The external object, `rpprocessor~`, was developed in C using the Max 5 API, in order to compensate for the shortcomings of the native audio buffers in Max/MSP. This object maintains the audio buffer and an internal representation of the RP, and reconstructs the output signal according to equation (9). In addition, `rpprocessor~` manages the creation, loading, saving and display of the RP data.

4.2. User interface

The user interface for the Max patch is shown in Figure 6. The user draws the desired recurrence pattern in the large grid area marked ①, by clicking/toggling cells between black and white. Once again, note that black squares indicate a recurrence. To maintain the standard topology of RPs, a non-editable main diagonal is added by default. Additionally, symmetry is enforced by automatically toggling equivalent cells the other side of the diagonal. The “clear” button above the RP deactivates all the cells in the block except those in the main diagonal.

The sliders marked ② allow the user to set N , the size of the RP, and L , the duration of each cell in the RP. Both of these values can be set using the sliders or the associated number boxes. Adjusting N changes the resolution of the RP grid, with higher

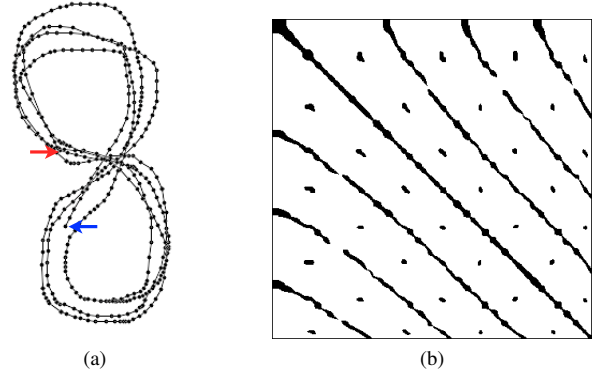


Figure 7: Constructing an RP from gestural data: (a) captured hand movements (the upper arrow and the lower arrow indicate the start point and the end point respectively) (b) RP generated from the gesture data shown in (a).

values corresponding to increased resolution. The segment duration is displayed both in milliseconds and samples.

The value of L is also displayed in beats per minute (bpm) in the “bpm” number box. According to equation (7), the value of \hat{n} varies between 1 and N , and represents a particular column of the RP. Because the duration of each cell in the RP is L , \hat{n} advances at a rate determined by L . We can therefore assume \hat{n} to represent a beat, and compute the value of L in bpm as follows: $bpm = 60/L$ (where L is measured in seconds).

The sliders in ③ are used to adjust the mix between the dry and wet signals and the overall gain. To enable the system, the user presses the large button in this region of the UI, thus turning on the analog-to-digital and digital-to-analog converters. Once the system has been enabled and is in play mode, the columns of the RP are highlighted in sequence, with the cells corresponding to the current \hat{n} in a given column colored red and the unactivated cells colored yellow. The highlighting proceeds from left to right, wrapping back to the first column once \hat{n} reaches N . As discussed above, \hat{n} is incremented every L seconds. The highlighting thus makes the value of L explicit, allowing the user to more easily synchronize his or her playing with the system.

The user can save the current RP into a text file by using the “write” button in ④. Pressing this button brings up a file chooser window, allowing the user to specify the desired name and location of the file. Pressing the “read” button also brings up a file chooser, allowing the user to load an RP from an existing text file. Once loaded, the RP will be displayed in the editing region (① in figure 6).

5. GENERATING AN RP FROM GESTURAL DATA

In the previous section, we have discussed how the user can draw a recurrence plot to control a time-variant delay line. However, because each RP is inherently associated with a time series, we have the ability to use any time series to control the audio effect. In this section, we provide a simple extension to our system in which we generate RPs from hand gestures.

To capture hand movements, we use an infrared LED and an Apple iSight web camera. The camera is fitted with a filter that blocks all visible light letting only infrared light pass. To capture

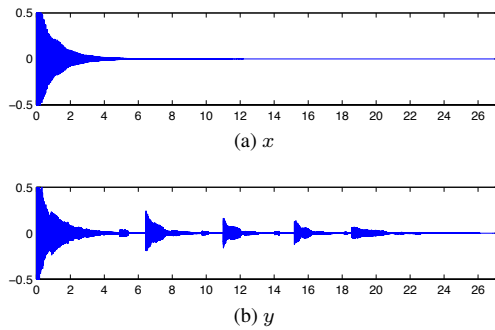


Figure 8: Real-time reconstruction example: (a) an input signal (b) a resulting sound using the RP in Figure 7b.

a gesture, the user holds the infrared LED in his or her hand, and moves it in view of the camera. The resulting X-Y positions of the LED, which we will define as $P(n) = (X_n, Y_n)$, are recorded at fixed time intervals.

Figure 7a shows a 400-point example time series of captured hand gestures, showing four repetitions of a figure-8 pattern. Positions are represented with black dots, with the starting point marked with a red arrow, and the end point marked with a blue one. The example trajectory illustrates an important issue with this approach. At the center of the figure-8 pattern the four upward and four downward trajectories pass through a small region of the 2-D space. According to equation (3), the proximity between these points makes them recurrences of each other. However, it can be argued that the upward and downward sub-trajectories are different enough that they should be regarded as different, and that the closeness between their constituent points is circumstantial. This implies measuring recurrences between sub-trajectories instead of between individual points, in order to avoid such spurious detections (and the resulting noisy plots). This can be done using a technique known as time-delay embedding [1, 2], where the n^{th} point of the embedded time-series is defined as:

$$\hat{P}(n) = (X_n, X_{n-1}, \dots, X_{n-\omega+1}, Y_n, Y_{n-1}, \dots, Y_{n-\omega+1}) \quad (10)$$

where $\omega \in \mathbb{N}$ is known as the embedding dimension (note that we assume the embedding delay to be 1). The RP can be obtained as:

$$R_{i,j} = \begin{cases} 1, & \|\hat{P}(i) - \hat{P}(j)\| < \epsilon \\ 0, & \|\hat{P}(i) - \hat{P}(j)\| \geq \epsilon \end{cases} \quad (11)$$

where, as before, ϵ is the margin of error and $\|\cdot\|$ is the Euclidean norm.

Figure 7b shows the RP computed from the gesture data using $\epsilon = 50$ and $\omega = 3$. It has three diagonal lines in the upper and lower triangular parts of the plot, indicating that the main diagonal recurs three times. The shorter diagonals in the RP correspond to the crossing of the sub-trajectories discussed above. They can be filtered out entirely with a larger ω value, however at the cost of missing recurrences in the larger diagonals. Finding an optimal parameterization for embedding is an active area of research [2].

Figure 8 shows the result of applying the RP from Figure 7b to an input signal x , with $L = 46.5\text{ms}$ (i.e., 2048 samples in 44.1 kHz sample rate). Sound examples, including stutter and time shuffling effects, as well as the sounds described above, are available at <http://marl.smusic.nyu.edu/rpprocess>.

6. CONCLUSIONS AND FUTURE WORK

We have presented a method for transforming an input signal based on the patterns of recurrence represented in an RP, and a realtime implementation of this method in which the user creates the RP using a graphical interface. The output signal from this system exhibits the recurring structures described by the RP. Unlike existing systems, however, the graphical interface in our system allows a user to easily experiment with different delay patterns. The use of gestural data to control the system suggests another intuitive means of controlling the delay parameters.

While the realtime system achieved the goal of producing complex delay effects, its time varying nature made it somewhat difficult to use in practice. In contrast to the case of a standard delay line, in which a user can easily synchronize his or her playing with the repeats, the time dependent behavior of our system made such synchronization difficult. The highlighting of the current column of the RP, while intended to ameliorate this problem, proved to be of limited use. Although further practice with the system would likely make synchronization easier, other remedies could include the use of an onset detector to trigger the start of an RP cycle.

We are also interested in more fully exploring the generation of the RP through gestural data. We feel that with these improvements, our system could prove to be a useful tool for performers and composers interested in producing complex delay effects.

7. REFERENCES

- [1] J.P. Eckmann, S.O. Kamphorst, and D. Ruelle, "Recurrence plots of dynamical systems," *EPL (Europhysics Letters)*, vol. 4, pp. 973, 1987.
- [2] N. Marwan, N. Wessel, U. Meyerfeldt, A. Schirdewan, and J. Kurths, "Recurrence plot based measures of complexity and its application to heart rate variability data," *Physics Reports*, vol. 438, no. 5 - 6, pp. 237 - 329, 2002.
- [3] J. D. Reiss and M. B. Sandler, "Nonlinear time series analysis of musical signals," in *Proceedings of the 6th International Conference on Digital Audio Effects (DAFX-03)*, 2003.
- [4] J. Serrá, X. Serra, and R. G. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, art. 093017, September 2009.
- [5] J. P. Bello, "Measuring structural similarity in music," *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.
- [6] M. Witten, "The sounds of science: II. Listening to dynamical systems—Towards a musical exploration of complexity," *Computers and Mathematics with Applications*, vol. 32, no. 1, pp. 145 - 173, 1996.
- [7] U. Zölzer (Ed.), *DAFX: digital audio effects*, John Wiley & Sons Inc, 1st edition, 2002.
- [8] D. Overholt, "Control of Signal Processing Algorithms using the MATRIX Interface," in *Audio Engineering Society 114th Convention*, 2003.
- [9] N. Collins, "Kid A, Amnesiac, and Hail to the Thief (review)," *Computer Music Journal*, vol. 4, no. 1, 2004.
- [10] N. Collins, "BBCut2: Integrating beat tracking and on-the-fly event analysis," *Journal of New Music Research*, vol. 35, no. 1, pp. 63-70, 2006.

A SOUND LOCALIZATION BASED INTERFACE FOR REAL-TIME CONTROL OF AUDIO PROCESSING

Daniele Salvati

AVIRES Lab.
Dep. of Mathematics and Computer Science
University of Udine, Italy
daniele.salvati@uniud.it

Sergio Canazza

Sound and Music Computing Group
Dep. of Information Engineering
University of Padova, Italy
canazza@dei.unipd.it

Antonio Rodà

Sound and Music Computing Group
Dep. of Information Engineering
University of Padova, Italy
roda@dei.unipd.it

ABSTRACT

This paper describes the implementation of an innovative musical interface based on the sound localization capability of a microphone array. Our proposal is to allow a musician to plan and conduct the expressivity of a performance, by controlling in real-time an audio processing module through the spatial movement of a sound source, i.e. voice, traditional musical instruments, sounding mobile devices. The proposed interface is able to locate and track the sound in a two-dimensional space with accuracy, so that the x-y coordinates of the sound source can be used to control the processing parameters. In particular, the paper is focused on the localization and tracking of harmonic sound sources in real moderate reverberant and noisy environment. To this purpose, we designed a system based on adaptive parameterized Generalized Cross-Correlation (GCC) and Phase Transform (PHAT) weighting with Zero-Crossing Rate (ZCR) threshold, a Wiener filter to improve the Signal to Noise Ratio (SNR) and a Kalman filter to make the position estimation more robust and accurate. We developed a Max/MSP external objects to test the system in a real scenario and to validate its usability.

1. INTRODUCTION

Music interaction is an important and new area in the field of audio based Human Computer Interaction (HCI) systems. The development of new interfaces for musical applications has the potential to change and enhance the experience of musical performance and in particular to allow a performer the interaction with a computer for real-time audio processing. The development of digital audio effects has always stimulated the design of interfaces for controlling the processing parameters. A large number of musical interfaces [1] has been implemented and tested with the goal of providing tools for gestural interaction with digital sounds.

In [2], the author divides gestural controllers into four main categories: gestural interfaces played by touching or holding the instrument, interfaces with haptic feedback, interfaces worn on the body and interfaces that may be played without any physical contact. In this last category, the position of the body might be used

without the need for the performer to wear or touch any special devices. Examples of such interfaces are: Gesture Wall [3], that uses electric field sensors to measure the position and movement of the player's hands and body in front of a projection screen; Litefoot [4], based on optical sensor; an interface based on video camera that allows the performers to use their full-body for controlling in real-time the generation of an expressive audio-visual feedback [5].

Musical interfaces are often used to allow the performer to enhance the expressive control on the sounds generated by their acoustic instruments in a live electronics context. E.g., in *Medea* by Adriano Guarnieri (2002) the movement of the bell of a trombone is captured by a camera [6] and mapped into parameters for sound spatialization; in *fili bianco-velati* (Guarnieri, 2005), the movement of a violinist is followed by a motion capture system based on infrared cameras.

In general, those kind of systems have considerable complexity and in some situations some problems. In fact, the performer has to wear sensors or devices which can be a hindrance to his/her movements; besides, in the camera-based systems there could be problems with the low and/or not always controllable lighting of the concert hall.

This paper describes the implementation of an innovative musical interface based on the sound localization capability of a microphone array. The interface allows a musician to plan and conduct the expressivity of a performance, by controlling in real-time an audio processing module through the spatial movement of a sound source. In this way a musician, during the performance, is able to interact with the live electronics system through the movement of his/her own musical instrument with an immediate, instinctive and gestural approach. The proposed interface is completely non-invasive (no need for markers, sensors or wires on the musician) and requires no dedicated hardware.

The system uses an algorithm based on an estimate of the Time Difference Of Arrival (TDOA) for sound source localization. Typically, these algorithms tend to reduce their performance in presence of competing sources, high reverberant environment, or low signal to noise ratio. Moreover, in the context of live electronics it is not always possible to have a controlled acoustic scene (there

could be other sources or noise due to the return of audio monitor), thus, we propose an innovative approach that combines an array of supercardioid polar pattern microphones (instead of the classic omnidirectional ones which are usually used in array processing) and a localization process task based on adaptive parameterized GCC-PHAT with ZCR threshold, a Wiener filter to improve the SNR and a Kalman filter to make the position estimation more robust and accurate.

The paper is organized as follow: Section 2 presents the system architecture, both the hardware and software aspects; the algorithms for the sound source localization are detailed in Section 3; finally, Section 4 shows some preliminary results.

2. SYSTEM ARCHITECTURE

The interface consists of three main components: i) a microphone array for signal acquisition; ii) signal processing algorithms for robust sound localization; iii) a mapping strategy to control the audio processing parameters.

The array is composed by three microphones arranged in a uniform linear placement. In this way we can localize a sound source in a plane (three microphones are the bare minimum). Signal processing algorithms estimate the the sound source position in a horizontal plane by providing its Cartesian coordinates. The last component implements the mapping strategy [7], so that the x-y coordinates are associated with audio processing parameters. For the purpose of testing, in this paper we have limited ourselves to explore a one-to-one mapping strategy, by using the x-y values to directly control two parameters of an audio effect, e.g. cutoff frequency and resonance of a filter or amount and decay time of a reverb. Of course, this task is closely related to user needs, and in literature there are a lot of works proposing strategies to transform from two-to-many parameters [8] [9] [10] [11] [12]. This paper is mainly focused on the localization task.

Figure 1 summarizes the system architecture. Sound source localization allows to extract information about the location of one or more sources using microphone arrays and signal processing techniques. A widely used approach to estimate the source position consists of two steps: in the first step, a set of TDOAs are estimated using measurements across various combinations of microphones; in the second step, knowing the position of sensors and the velocity of sound, the source position is calculated by means of geometric constraints and using approximation methods such as least-square techniques [13]. The traditional technique to estimate the TDOA between a pair of microphones is the GCC-PHAT [14]. It is highly effective in a moderately reverberant and noisy environment. Unfortunately, concerning musical sounds that are mainly harmonics, the GCC-PHAT approach does not work, because the PHAT filter normalizes the GCC according to the spectrum magnitude. Then, new considerations are required to estimate the TDOA for pseudo-periodic signals. Our proposal is to use a parameterized GCC-PHAT, that weights the contribution of the PHAT filtering, depending on the threshold of the ZCR parameters.

A de-noise algorithm based on Wiener filter is used to improve the SNR of the signals. When the maximum peak detection does not observe any source, it is computed an average estimation of noise (noise print), which will be subtracted in all three signals before the TDOA estimation task.

Then, starting from the estimated TDOA between microphones $\hat{\tau}_{12}$ and $\hat{\tau}_{23}$, it is possible to calculate the coordinates of the source by means of geometric constraints. In a near-field environment we

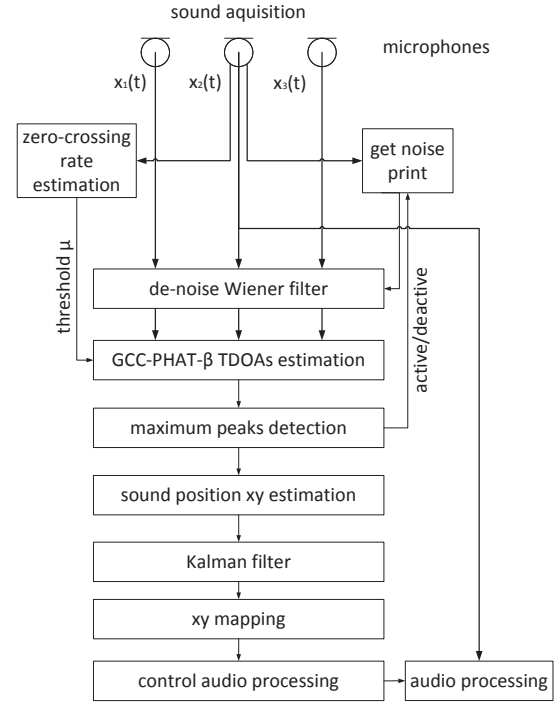


Figure 1: Block diagram of interface.

have

$$\hat{x} = r \cos(\theta) \quad (1)$$

$$\hat{y} = r \sin(\theta) \quad (2)$$

where the axis origin is placed in microphone 2, r is the distance between the sound source and the microphone 2, and θ is the angle between r and the x axis

$$\theta = \arccos\left(\frac{c(\hat{\tau}_{12} + \hat{\tau}_{23})(\hat{\tau}_{12}\hat{\tau}_{23}c^2 - d^2)}{d(2d^2 - c^2(\hat{\tau}_{12}^2 + \hat{\tau}_{23}^2))}\right) \quad (3)$$

$$r = \frac{\hat{\tau}_{12}^2 c^2 - d^2}{2(\hat{\tau}_{12}c + d \cos \theta)} \quad (4)$$

where c is speed of sound and d is the distance between microphones.

Finally, a second filter provides a more accurate tracking of the source position, by means of the Kalman theory. The Kalman filter is able to provide an estimation of the position of the source, also if the TDOA estimation task misses the target in some frame of analysis.

3. SOUND SOURCE LOCALIZATION

3.1. TDOA estimation using GCC-PHAT

GCC [14] is the classic method to estimate the relative time delay associated with the acoustic signals received by a pair of microphones in a moderate reverberant and noisy environment. It basically consists of a cross-correlation followed by a filter that aims at reducing the performance degradation due to additive noise and

multi-path channel effects. The signals received at the two microphones $x_1(t)$ and $x_2(t)$ may be modeled as

$$\begin{aligned} x_1(t) &= h_1(t) * s(t) + n_1(t) \\ x_2(t) &= h_2(t) * s(t - \tau) + n_2(t) \end{aligned} \quad (5)$$

where τ is the relative signal delay of interest, $h_1(t)$ and $h_2(t)$ represent the impulse responses of the reverberant channels, $s(t)$ is the sound signal, $n_1(t)$ and $n_2(t)$ correspond to uncorrelated noise, and $*$ denotes linear convolution. The GCC in the frequency domain is

$$R_{x_1 x_2}(t) = \sum_{w=0}^{L-1} \Psi(w) S_{x_1 x_2}(w) e^{\frac{jw t}{L}} \quad (6)$$

where w is the frequency index, L is the number of samples of the observation time, $\Psi(w)$ is the frequency domain weighting function, and the cross-spectrum of the two signals is defined as

$$S_{x_1 x_2}(w) = E\{X_1(w) X_2^*(w)\} \quad (7)$$

where $X_1(w)$ and $X_2(w)$ are the Discrete Fourier Transform (DFT) of the signals and $*$ denotes the complex conjugate. GCC is used for minimizing the influence of moderate uncorrelated noise and moderate multipath interference, maximizing the peak in correspondence of the time delay.

The relative time delay τ is obtained by an estimation of the maximum peak detection in the filter cross-correlation function

$$\hat{\tau} = \underset{t}{\operatorname{argmin}} R_{x_1 x_2}(t). \quad (8)$$

PHAT [14] weighting is the traditional and most used function. It places equal importance on each frequency by dividing the spectrum by its magnitude. It was later shown that it is more robust and reliable in realistic reverberant conditions than other weighting functions designed to be statistically optimal under specific nonreverberant noise conditions [15]. The PHAT weighting function normalizes the amplitude of the spectral density of the two signals and uses only the phase information to compute the GCC

$$\Psi_{\text{PHAT}}(w) = \frac{1}{|S_{x_1 x_2}(w)|}. \quad (9)$$

It is widely acknowledged that GCC performance is dramatically reduced in case of harmonic sound, or generally pseudo-periodic sounds. In fact, the GCC have less capability to reduce the deleterious effects of noise and reverberation, when it is applied to pseudo-periodic sound.

3.2. Adaptive parameterized GCC-PHAT with zero-crossing rate threshold

The PHAT weighting can be generalized to parametrically control the level of influence from the magnitude spectrum [16]. This transform will be referred to as the PHAT- β and defined as

$$\Psi_{\text{PHAT}-\beta}(w) = \frac{1}{|S_{x_1 x_2}(w)|^\beta} \quad (10)$$

where β varies between 0 and 1. When $\beta = 1$, equation (10) becomes the conventional PHAT and the modulus of the Fourier transform becomes 1 for all frequencies, when $\beta = 0$ the PHAT has no effect on the original signal, and we have the cross-correlation function.

Therefore, in case of harmonic sounds we can use an intermediate value of β so that we can detect the peak to estimate the time delay between signals, and to have a system, at least in part, which exploits the benefits of PHAT filtering to improve performance in a moderately reverberant and noisy environments. To adapt the value of β we use the ZCR to check if sound source is periodic or not. ZCR is a very useful audio feature, and it is defined as the number of times that the audio waveform crosses the zero axis

$$\text{ZCR}(t) = \frac{1}{2N} \sum_{i=1}^N |\operatorname{sgn}(x(t+i)) - \operatorname{sgn}(x(t+i-1))|. \quad (11)$$

where $\operatorname{sgn}(x)$ is the sign function.

Then, we can express the adaptive parametrized GCC-PHAT, identifying by experimental tests a suitable threshold μ such as

$$\begin{cases} \beta = 1, & \text{if } \text{ZCR} \geq \mu \\ \beta < 1, & \text{if } \text{ZCR} < \mu \end{cases} \quad (12)$$

3.3. De-noise Wiener filter

Frequency domain methods, which are based on the Short Time Spectral Attenuation (STSA) [17], require a little information to carry out the filtering (*a priori* information): only an estimate of the noise present is necessary (noise print), since it is assumed to be stationary along the entire signal. Any further information needed (*a posteriori* information) is automatically calculated by the algorithm through the analysis of the characteristics of the signal. Since this method is easy to use and is generally applied to different typologies of audio signals, they are employed in commercial hardware and software systems.

These de-noise systems consist of two important components: a noise estimation method and a suppression rule. These techniques employ a signal analysis through the Short-Time Fourier Transform (STFT) (which is calculated on windowed section of the signal as it changes over time) and can be considered as a non-stationary adaptation of the Wiener filter [18] in the frequency domain. In particular, Short Time Spectral Attenuation (STSA) consists in applying the short-time spectrum of the noise to a time-varying suppression and does not require the definition of a model for the audio signal. Suppose considering the useful signal $s(t)$ as a stationary aleatory process to which some noise $n(t)$ is added (uncorrelated with $x(t)$) to produce the degraded signal $x(t)$. The relation that connects the respective power spectral densities is therefore

$$P_x(w) = P_s(w) + P_n(w). \quad (13)$$

If we hypothesize to succeed in retrieving an adequate estimate of $P_n(w)$, during the silence intervals of the signal $x(t)$, and in the musical portions of $P_x(w)$, we can expect to obtain an estimate of the spectrum of $s(t)$ by subtracting $P_n(w)$ from $P_x(w)$; the initial assumption of stationariness can be considered locally satisfied since short temporal windows are employed. Note that the use of a short-time signal analysis is equivalent to the use of a filter bank. First each channel (that is, the output of each filter) is appropriately attenuated and then it is possible to proceed with the synthesis of the restored signal. The timevarying attenuation applied to each channel is calculated through a determined suppression rule, which has the purpose to produce an estimate (for each channel) of the noise power. Each particular STSA technique is characterized by the implementation of the filter bank and of the suppression rule.

If we denote the STFT of the $x(t)$ noisy signal with $X(t, w_k)$, where t represents the temporal index and w_k the frequency index (with $k = 1 \dots N$, N represents the number of STFT channels), the result of the suppressing rule application can be interpreted as the application of a $G(t, w_k)$ gain to each value $Y(t, w_k)$ of the STFT of the noisy signal. This gain corresponds to a signal attenuation and is included between 0 and 1. In most of the suppression rules, $G(t, w_k)$ only depends on the noisy signal power level (measured at the same point) and on the estimate of the noisy power at the w_k frequency

$$\hat{P}_n(w_k) = E\{|N(t, w_k)|^2\} \quad (14)$$

(which does not depend on the temporal index t due to the presumed noise stationariness). At this point a *relative* signal can be defined

$$Q(t, w_k) = \frac{|X(t, w_k)|^2}{\hat{P}_n(w_k)} \quad (15)$$

which, starting from the hypothesis that the $n(t)$ noise is not correlated to the $x(t)$ signal, we deduce should be greater than 1

$$E\{Q(t, w_k)\} = 1 + \frac{E\{|S(t, w_k)|^2\}}{\hat{P}_n(w_k)}. \quad (16)$$

A typical suppression rule is based on the Wiener filter [18] and can be formulated as follows

$$G(t, w_k) = \frac{|X(t, w_k)|^2 - \hat{P}_n(w_k)}{|X(t, w_k)|^2}. \quad (17)$$

3.4. Kalman filter

The Kalman filter [19] is the optimal recursive Bayesian filter for linear systems observed in the presence of Gaussian noise. We consider that the state of the sound localization could be summarized by two position variables, x and y , and two velocities, v_x and v_y . These four variables are the elements of the state vector \mathbf{x}_t

$$\mathbf{x}_t = [x, y, v_x, v_y]^T. \quad (18)$$

The process model relates the state at a previous time $t - 1$ with the current state at time t , so we can write

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{w}_{t-1} \quad (19)$$

where \mathbf{F} is the transfer matrix and \mathbf{w}_{t-1} is the process noise associated with random events or forces that directly affect the actual state of the system. We assume that the components of \mathbf{w}_{t-1} have Gaussian distribution with zero mean normal distribution with covariance matrix \mathbf{Q}_t , $\mathbf{w}_{t-1} \sim N(0, \mathbf{Q}_t)$. Considering the dynamical motion, if we measured the system to be at position x with some velocity v at time t , then at time $t + dt$ we would expect the system to be located at position $x + v \cdot dt$, thus this suggests that the correct form for \mathbf{F} is

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (20)$$

At time t an observation \mathbf{z}_t of the true state \mathbf{x}_t is made according to the measurement model

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t \quad (21)$$

where \mathbf{H} is the observation model which maps the true state space into the observed space and \mathbf{v}_t is the observation noise which is assumed to be zero mean Gaussian white noise with covariance \mathbf{R}_t , $\mathbf{v}_t \sim N(0, \mathbf{R}_t)$. We only measure the position variables. Hence, we have

$$\mathbf{z}_t = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}, \quad (22)$$

and then we have

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \quad (23)$$

The filter equations can be divided into a prediction and a correction step. The prediction step projects forward the current state and covariance to obtain an a priori estimate. After that the correction step uses a new measurement to get an improved a posteriori estimate. In prediction step the time update equations are

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1}, \quad (24)$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^T + \mathbf{Q}_{t-1}, \quad (25)$$

where \mathbf{P}_t denotes the error covariance matrix. In the correction step the measurement update equations are

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1}), \quad (26)$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}) \mathbf{P}_{t|t-1}, \quad (27)$$

where \mathbf{I} is the identity matrix and so-called Kalman gain matrix is

$$\mathbf{K}_t = \mathbf{P}_{t-1|t-1} \mathbf{H}^T (\mathbf{H} \mathbf{P}_{t-1|t-1} \mathbf{H}^T + \mathbf{R}_t)^{-1}. \quad (28)$$

This formulation requires that the dynamic of the system is linear. However our specific problem is non-linear. To accommodate non-linear state transition and observation models, the Extended Kalman Filter (EKF) [20] implements a local linearization of the models. Thus, we need to compute new values for \mathbf{F} , at every time step, based on the state \mathbf{x} to approximate the real update.

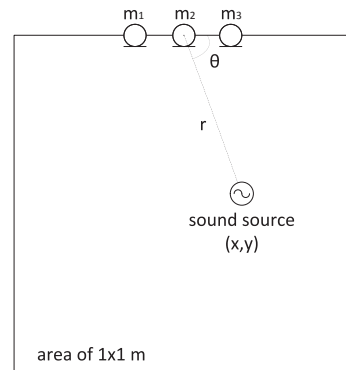


Figure 2: The map of the considered control area.

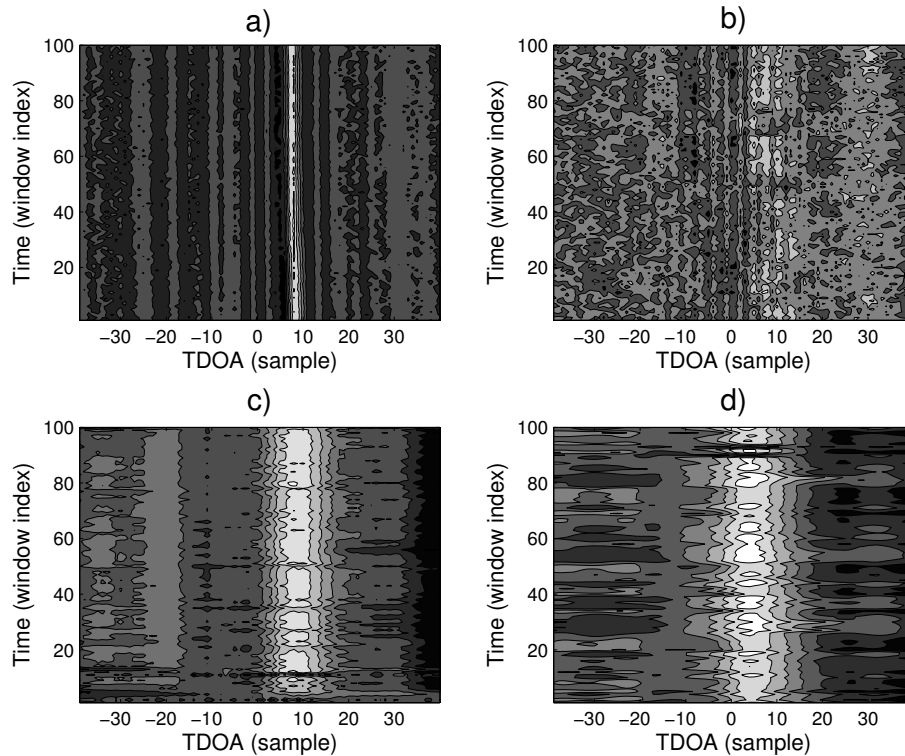


Figure 3: Comparison of parameterized PHAT- β TDOA estimation performance. All sound sources are approximately located in (0,50) cm. a) White noise played on mobile device, $\beta = 1$. b) Flute, $\beta = 1$. c) Flute, $\beta = 0.65$. d) Flute, $\beta = 0.65$ and de-noise Wiener filter. The variances of TDOA estimation are: a) $\sigma_a^2 = 0.04$; b) $\sigma_b^2 = 80$; c) $\sigma_c^2 = 0.65$; d) $\sigma_d^2 = 0.5$.

4. RESULTS

Some experimental results related to the localization performance of the interface in a real scenario are presented. To verify and validate our approach to the localization of pseudo-periodic sounds we used and compared three types of sources: white noise played on a mobile device, a flute played by a musician and a human voice. The interface works with sampling rate of 96 kHz, a Hanning analysis window of 42 ms, a time window for the estimation of the average noise (noise print) of 4.2 s. We used three microphones with supercardioid pickup pattern, which are the most frequently used microphones to acquire sound signals in electroacoustic music. It is important to highlight that the classic microphone for array processing is the omnidirectional polar pattern, but its use is not appropriate in this context because of possible interference of the loudspeakers during the application in live performance. However, as we shall see, the use of directional microphones allows the localization of an acoustic source in the small area of interest (Figure 2). The distance between microphones is $d = 15$ cm. The working area is included in a square of side 1 meter. The axis origin coincides with microphone 2 (m_2) position, and x axis can vary between -50 cm and 50 cm and y axis between 0 and 100 cm (Figure 2).

Experiments have been done in a rectangular room of 3.5×4.5 m, with a moderately reverberant and noisy environment. Figure 3 shows a comparison of parameterized PHAT- β TDOA estimation performance. We made four tests with different parameters of interface configuration. We consider the TDOA estimation between

microphone 2 and 3. All sound sources are approximately located in the center of interested map, (a) (5,52) cm, (b) (4,51) cm, (c) (5,53) cm, (d) (3,51) cm. In the first test (a), we played a continuous white noise signal by a mobile device with $\beta = 1$ interface configuration. In this way we checked the whole efficiency offered by the PHAT filter to optimize the TDOA estimation, reducing the degradation effects due to noise and reverberation. We can see in Figure 3 how the maximum peak detection is clearly visible (white line). We can also see the effects of multipath reverberation represented by the other parallel gray lines. The value of TDOA estimation is $\hat{\tau}_{23} = 7$ (sample). The variance of TDOA maximum peak during the whole reproduction of sound is $\sigma_a^2 = 0.04$. The TDOA estimation is extremely accurate. In test (b), is considered a flute again with $\beta = 1$ parameter. As expected, the source is not detected ($\sigma_b^2 = 80$). Subsequently, in test (c) we examined a flute with $\beta = 0.65$ setting. The source is detected as shown in Figure 3. The mean value of TDOA estimation is $\hat{\tau}_{23} = 7$ (sample) and it corresponds to the correct position of the source, the variance results $\sigma_c^2 = 0.65$. In the last test (d), we considered once more a flute with $\beta = 0.65$ and the de-noise Wiener filter task. The mean value of TDOA estimation is $\hat{\tau}_{23} = 5$ (sample), the variance results $\sigma_d^2 = 0.5$. Hence, in this case, a lower value of variance indicates a less swinging of the TDOA than the average value, which is the correct location of the source.

Therefore, the parameterized PHAT- β allows the TDOA estimation of harmonic sounds, and de-noise component can improve the accuracy. However, the comparison with test (a), whose robust and well-defined result we aim to obtain, does not give yet satisfac-

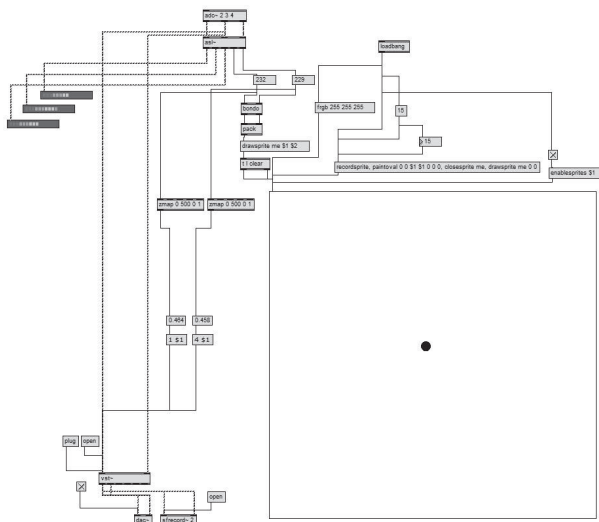


Figure 6: *The Max/MSP interface with the external object asl~.*

tory results. A parameterization of PHAT with value of $\beta = 0.65$, in according to [16], is a good compromise between filtering and detection.

Before considering the experiments with the Kalman filter, we show the results of a test with a human voice, ranging from harmonic to noise sounds, in order to verify the threshold value of zero-crossing rate for the activation of PHAT- β . The human voice source is located in (-20,70) cm. Figure 4 shows the results of the ZCR and adaptive parameterized GCC-PHAT- β with threshold value of $\mu = 0.03$, $\beta = 0.65$ when $ZCR < \mu$ and de-noise Wiener filter. We believe that this value μ is enough to achieve an adequate adaptation of GCC. Still in Figure 4, we can note that when the sound becomes harmonic, and then we have partially filtered GCC with PHAT, the TDOA peak tends to widen, reducing its robustness, but still allowing the estimation of source position.

Finally, the last test on the localization performance shows the effectiveness of the Kalman filter to make the xy coordinates more accurate and usable in the interface. Once again we used a flute moving within the mapped area. The threshold value of ZCR is $\mu = 0.03$, $\beta = 0.65$, and de-noise task is active. As you can see from Figure 5, the black lines, which represents the data after the Kalman filtering, are reported in order to have less stability problems due to reverberation. In fact, the estimated raw data (gray lines) present very high swinging values, which would make the interface inappropriate to control the processing parameters.

In conclusion, we implemented the system by developing a Max/MSP external object, named `asl~`, in order to validate the interface in real-world music application. The object receives incoming audio signals acquired by the three microphones and, as output, it gives the position of the sound source. The object performs all the signal processing techniques described in the previous sections. Moreover, a simple Max/MSP patch (see Figure 6) has been developed in order to control in real-time an audio processor. As mentioned, `xy` values have been used to directly control the parameters of an audio effect. We made use of different VST plug-ins, such as reverb and delay effects, with encouraging results.

5. CONCLUSIONS

We described a digital interface that incorporates real-time sound source localization for gestural control without any physical contact, which can be used as audio HCI system to enhance the experience of a musical performance. In order to work with harmonic sounds, we proposed a system consisting of adaptive parameterized GCC-PHAT with zero-crossing rate threshold. We have seen that this solution allows to locate sources such as musical instruments, but it is less robust in moderate reverberation and noisy environments, comparing to the standard GCC-PHAT. For this reason, we included two filters. The first one has been set up using the STSA with Wiener filter, before the TDOA estimation task, in order to improve the SNR of signals. The second one has been formulated using the Kalman filter theory, after the estimation of the source position. In this way, we obtained an accurate localization system. We used a linear array of three supercardioid polar pattern microphones, and we have seen that we are able to locate the sound source inside an area of one square meter. The usability of the interface was validated by developing a Max/MSP external object, so that we can map the xy position of the sound source (i.e. voice, traditional musical instruments and sounding mobile devices) into control parameters.

Future works include the test and use of the sound localization based interface in real application of live performance to verify how the system works with interfering sources from a sound reinforcement system and other instruments. In addition, we plan to test other mapping strategies in order to obtain a more articulate, complex and interesting system.

6. REFERENCES

- [1] E. R. Miranda and M. M. Wanderley, *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*, A-R Editions, 2006.
- [2] T. Todoroff, *DAFX: Digital Audio Effects*, chapter Control of Digital Audio Effects, pp. 465–498, Wiley, 2001.
- [3] J. Paradiso and N. Gershenfeld, “Musical applications of electric field sensing,” *Computer Music Journal*, vol. 21(3), pp. 69–89, 1997.
- [4] N. Griffith and M. Fernstrom, “Litefoot - a floor space for recording dance and controlling media,” in *Proc. International Computer Music Conference*, 1998, pp. 475–481.
- [5] G. Castellano, R. Bresin, A. Camurri, and G. Volpe, “Expressive control of music and visual media by full-body movement,” in *Proc. International Conference on New Interfaces for Musical Expression*, 2007, pp. 390–391.
- [6] A. de Götzen, “Enhancing engagement in multimodality environments by sound movements in a virtual space,” *IEEE Multimedia*, vol. 11, pp. 4–8, 2006.
- [7] V. Verfaillie, M. Wanderley, and P. Depalle, “Mapping strategies for gestural and adaptive control of digital audio effects,” *Journal of New Music Research*, vol. 35, pp. 71–93, 2006.
- [8] J. Allouis and J. Y. Bernier, “The SYTER project: Sound processor design and software overview,” in *Proc. International Computer Music Conference*, 1982, pp. 232–240.
- [9] M. Spain and R. Polfremam, “Interpolator: a twodimensional graphical interpolation system for the simultaneous control

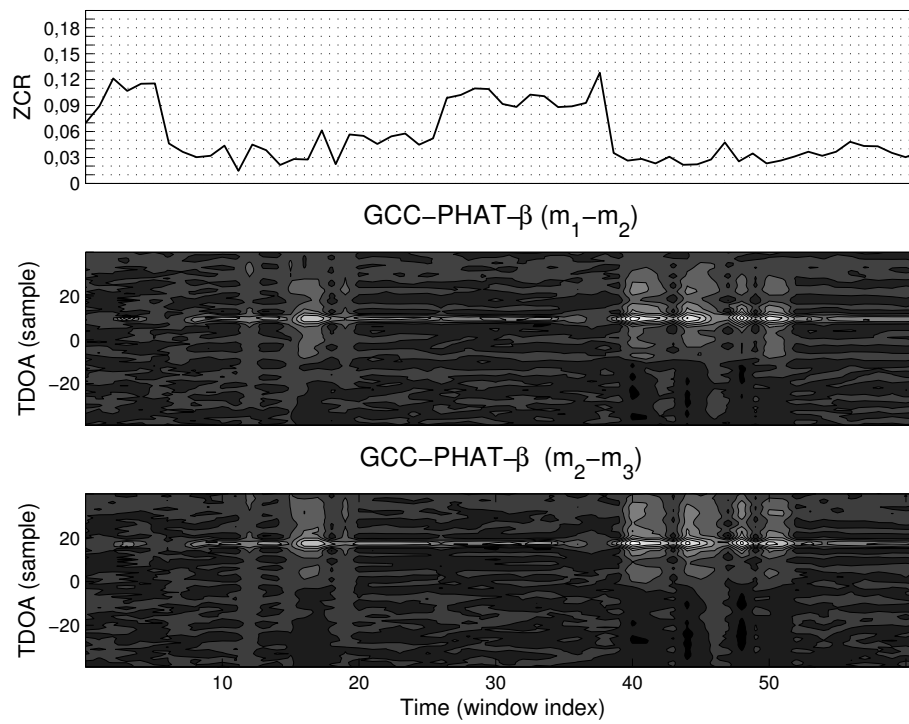


Figure 4: Human voice located (-20,70) cm. ZCR and parameterized GCC-PHAT- β with threshold value of $\mu = 0.03$; ($m_1 - m_2$) is referred to TDOA estimation between microphone 1 and 2, whereas ($m_2 - m_3$) between 2 and 3.

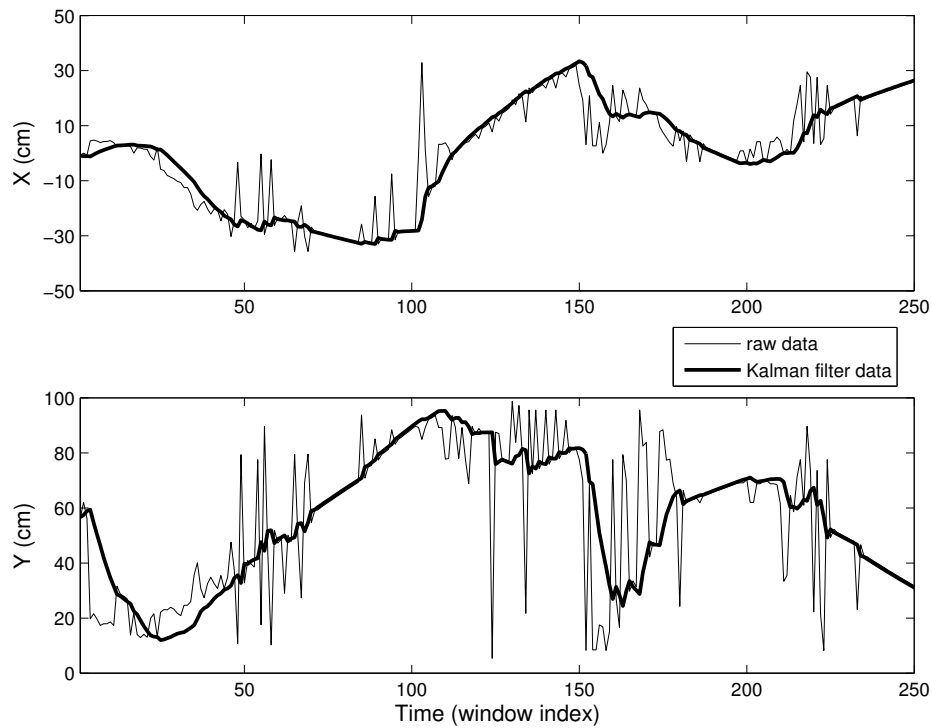


Figure 5: Flute performance moving within the interested area; X and Y position of Kalman filtering data (black lines) and raw data (gray lines).

- of digital signal processing parameters,” *Organised Sound*, vol. 6, no. 2, pp. 147–152, 2001.
- [10] A. Momeni and D. Wessel, “Characterizing and controlling musical material intuitively with geometric models,” in *Proc. International Conference on New Interfaces for Musical Expression*, 2003, pp. 54–62.
 - [11] R. Bencina, “The metasurface: applying natural neighbour interpolation to two-to-many mapping,” in *Proc. International Conference on New Interfaces for Musical Expression*, 2005, pp. 101–104.
 - [12] O. Larkin, “INTLIB - A Graphical Preset Interpolator For Max MSP,” in *Proc. International Computer Music Conference*, 2007.
 - [13] R. O. Schmidt, “A new approach to geometry of range difference location,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-8 Issue: 6, pp. 821–835, 1972.
 - [14] C. Knapp and G. Carter, “The generalized correlation method for estimation of time delay,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 24, no. 4, pp. 320–327, May 1976.
 - [15] M. Omologo and P. Svaizer, “Acoustic event localization using a crosspower-spectrum based technique,” in *Proc. IEEE ICASSP*, 1994, vol. 2, pp. 273–276.
 - [16] K. D. Donohue, J. Hannemann, and H. G. Dietz, “Performance of phase transform for detecting sound sources with microphone arrays in reverberant and noisy environments,” *Signal Processing*, vol. 87, pp. 1677–1691, July 2007.
 - [17] S. Boll, “Suppression of acoustic noise in speech using spectral subtraction,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-27, no. 2, pp. 113–120, 1979.
 - [18] N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series with engineering applications*, Cambridge, MIT Press, Massachusetts, 1949.
 - [19] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.
 - [20] S. Schmidt, “Applications of state-space methods to navigation problems,” *Advances in Control Systems*, vol. 3, pp. 293–340, 1966.

SIMILARITY-BASED SOUND SOURCE LOCALIZATION WITH A COINCIDENT MICROPHONE ARRAY

Karl Freiburger,^{*} [†]

Institute of Electronic Music and Acoustics,
University of Music and Performing Arts
Graz, Austria
karl.freiberger@gmx.at

Alois Sontacchi,[†]

Institute of Electronic Music and Acoustics,
University of Music and Performing Arts
Graz, Austria
sontacchi@iem.at

ABSTRACT

This paper presents a robust, accurate sound source localization method using a compact, near-coincident microphone array. We derive features by combining the microphone signals and determine the direction of a single sound source by similarity matching. Therefore, the observed features are compared with a set of previously measured reference features, which are stored in a look-up table. By proper processing in the similarity domain, we are able to deal with signal pauses and low SNR without the need of a separate detection algorithm. For practical evaluation, we made recordings of speech signals (both loudspeaker-playback and human speaker) with a planar 4-channel prototype array in a medium-sized room. The proposed approach clearly outperforms existing coincident localization methods. We achieve high accuracy (2° mean absolute azimuth error at 0 dB SNR) for static sources, while being able to quickly follow rapid source angle changes.

1. INTRODUCTION

The task of acoustic source localization (ASL) is to estimate the location of one or several sound sources given acoustic information only. Typically, a microphone array is used as a sensor front-end. ASL can be used to determine the steering direction of a microphone array beamformer and/or to direct a video camera towards the estimated source direction [1]. Typical applications are hands-free communication, conferencing systems and human-like robots. Most of the established methods for ASL use spatially distributed microphones to capture the direction-dependent time difference of arrival (TDOA) [2, 3]. Since the magnitude of the TDOA is directly related to the microphone spacing, arrays well suited for TDOA-based ASL require more space than so called near-coincident microphone arrays (NCMA).

NCMA consist of two or more microphone capsules having their acoustic center as close to each other as possible. Instead of evaluating time differences, the key principle behind ASL with NCMA is to use level differences between the microphone signals. These level differences can be caused either by dedicated directional capsules [4, 5, 6] or by omni-directional transducers which are differentially combined [7]. Established methods for coincident localization [4, 5, 6, 7] share in common that the source direction is determined by computing the active sound intensity vector.

In this paper, we propose a new coincident ASL-method based on supervised pattern recognition via a minimum distance classifier. The principle of our approach is depicted in Fig. 1. We derive specific features \mathbf{Y} from the captured microphone signals and compare them to a set of pre-measured reference features, stored in a look-up table. This table consists of a number Q of feature vectors $\mathbf{Y}(\Theta_q)$, each relating to a specific source position Θ_q . Basically, the source position is estimated as that position Θ_q where $\mathbf{Y}(\Theta_q)$ is most similar to \mathbf{Y} .

To be able to track changing source locations, the processing is performed frame-wise. For each frame l , we obtain a similarity curve (SC) $C_l(\Theta_q)$ via the Euclidean distance between \mathbf{Y}_l and $\mathbf{Y}(\Theta_q)$. The SC is ought to peak at the position of the sound source. If the shape of the SC is however flat, without a clear, global maximum, it is likely that the current frame would produce a more or less random source location estimate. This is for instance typical for a speaking pause between two words. By using the shape of the SC for weighting the influence of the observed frame with respect to previous ones, we can however effectively suppress the influence of signal pauses. With that, we obtain a stable source position estimate without jumping away from the source in signal pauses which is important in many practical applications such as camera- or beam-steering.

Instead of smoothing the sequence of location estimates with a fixed time-constant, our approach is adaptive. This makes our position estimator able to quickly follow a sudden change of the source location and produce a very smooth result without outliers in case of a static source position. In contrast to a separate voice activity detector, our SC-shape based detection method is independent of the signal type, e.g. speech, narrow-band, noise, transient signals, and comes at virtually no additional computational cost.

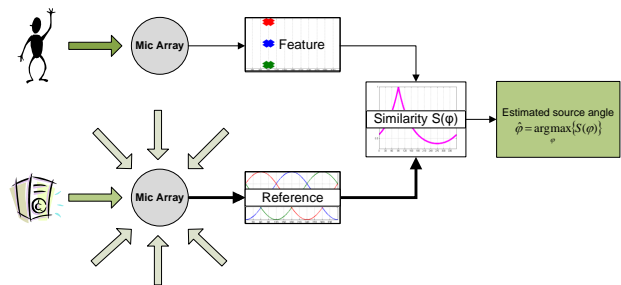


Figure 1: Supervised pattern classification principle.

^{*} New affiliation: BCT - Electronic GesmbH, Saalachstrasse 88, 5020 Salzburg, Austria

[†] Thanks to AKG Acoustics, GmbH, Vienna, esp. Martin Opitz, Marco Riemann and Matthias Maly-Persy, for providing the prototype microphone array and supporting our work.

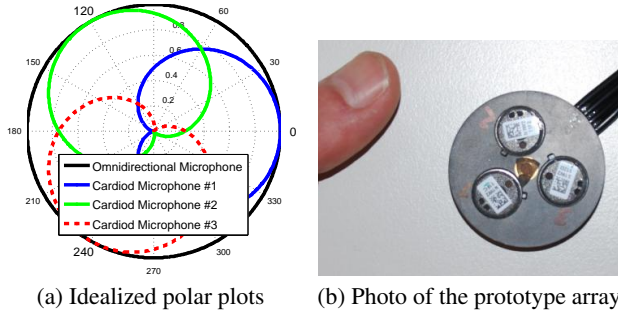


Figure 2: The microphone array used in this paper.

A good basis for localization of a single sound source is to track strong frequencies and compute an average SC over those frequencies. To extend our method to tracking of multiple sources, we suggest to perform clustering in the frequency-dependent SC instead of averaging. Multi-source tracking is however beyond the scope of this paper.

A basic property inherent to coincident localization is that two sources providing energy at the same frequency at the same time cannot be separated. Instead, if the sources have equal level, the mean direction is detected. However, multiple speakers are typically not always active at the exact same time and frequency and hence histogram or time-averaging approaches make multi-speaker localization possible [7].

In [7], it is suggested to use a measure of diffuseness as a reliability measure, which is conceptually very similar to our approach with rating the shape of the SC. As outlined above, we do however not use a threshold and do not use a fixed time constant for averaging, which makes our position estimator able to react either quick or smooth, depending on the situation. Another point relevant in practice, is the effect of microphone mismatch. Here, the coincident localization approach is likely to get less accurate, will however not completely fail, because the the overall characteristic of the level differences will not completely change due to manufacturing tolerances.

2. METHOD

2.1. Signal Model

Consider a single, acoustic point source s located at a position $\Theta_s = [\varphi_s, \vartheta_s, r_s]^T$. φ , ϑ , and r denote the spherical coordinates azimuth, elevation and radius, respectively. The coordinate system is centered at the center of a coincident microphone array. This array consists of M microphone capsules indexed by $m = 0, \dots, M-1$. Our goal is to estimate Θ_s from the microphone array signals. In short time Fourier transform (STFT) domain, a linear, time invariant (LTI) model of the m^{th} microphone signal is given as

$$X_m(l, f) = S(l, f) \cdot H_{m|\Theta_s}(l, f) + V_m(l, f) \quad (1)$$

where $X_m(l, f)$, $S(l, f)$ and $V_m(l, f)$ represent the m^{th} microphone, acoustic source and an additive disturbance (noise) signal, respectively. l is the frame time index and f the frequency index. $H_{m|\Theta_s}(l, f)$ represents the frequency response of the m^{th} microphone given a source position Θ_s .

Because the frequency response is dependent on the source position, we refer to $H_{m|\Theta_s}(l, f)$ as the position dependent frequency response (PDFR). It models frequency dependence as well as the directivity and the proximity effect [8] of the microphone. The PDFR of an ideal first order microphone including the proximity effect, is given as [8]

$$H_{m|\Theta_s}(f) = (1 - \beta_m) + \beta_m \cos(\varphi - \varphi_m) \cos(\vartheta - \vartheta_m) \frac{1 + j \frac{2\pi f}{c} r}{j \frac{2\pi f}{c} r} \quad (2)$$

where $j = \sqrt{-1}$, c is the speed of sound, (φ_m, ϑ_m) is the look-direction of the microphone and β_m specifies the directivity. For high $2\pi f/c \cdot r$ (far-field), $\beta_m = 0.5$, $\beta_m = 0$, $\beta_m = 1$ yields the well-known cardioid, omni-directional and figure-8 polar pattern, respectively. To model a real microphone, the PDFR can be obtained by means of impulse response measurements, e.g. using the exponential sine sweep method [9].

2.2. Microphone array

For the following discussion and derivation of our ASL-algorithm we restrict to the planar, 4-channel NCMA configuration depicted in Fig. 2. This array consists of one omni-directional microphone capsule and three directional, first order microphones (cardioid polar pattern) respectively. The cardioids are oriented towards the azimuth angles 0° , 120° and 240° , respectively, within the same plane $\vartheta = 0$. Instead of using a separate omni-directional capsule, the omni-characteristic can also be achieved by summing the cardioids [10]. In our experiments, the source localization performance was the same in both cases. Using a separate microphone can however produce a better low-end sound when coincident beamsteering is performed.

Due to the planar setup, robust estimation of the elevation angle ϑ_s is hardly possible. With a 3D-array such as the SFM it should however be possible to perform estimation of the elevation angle equally well as azimuth-estimation. The proximity effect provides a physical basis for estimation of the source distance r_s . However, first experiments and theoretical considerations indicate, that distance estimation is very sensitive to noise and limited to close (nearfield) sources [10]. Therefore, this paper restricts to tracking of the source azimuth angle. Hence, we use φ instead of Θ in (4) and all following equations. Furthermore, only tracking of a single sound source is considered. We do however suggest how to extend the presented method to allow for tracking of multiple sources at the same time.

2.3. Features

The basic idea behind our features is that there is a direction-dependent triplet of cardioid microphone gains (cf. Fig. 2a and Fig. 3, top). For our observed feature vector $\mathbf{Y}(l, f)$, we must try to obtain to these gains from the microphone signals $X_m(l, f)$.

$$\mathbf{Y}(l, f) = [Y_1(l, f), Y_2(l, f), Y_3(l, f)]^T \quad (3)$$

$$Y_m(l, f) = \frac{|X_m(l, f)|}{|X_0(l, f)|} = \frac{|S(l, f) \cdot H_{m|\varphi_s}(l, f) + V_m(l, f)|}{|S(l, f) \cdot H_{0|\varphi_s}(l, f) + V_0(l, f)|} \quad (4)$$

The directional microphones are indexed by $m = 1, 2, 3$, and $m = 0$ is the omni-directional channel. The normalization by

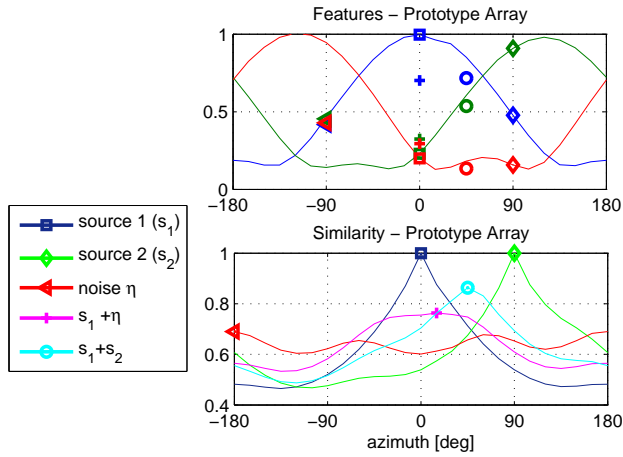


Figure 3: Effect of different source configurations (single sources, omni-directional noise, mix). Top: Reference features (solid lines) and feature vectors (markers) on basis of measured data of the prototype array (1 kHz, $r = 1$ m, $\vartheta = 0$). Bottom: Corresponding similarity curves (SCs). In case of a single sound source there is a sharp peak in the SC. If several directions contribute energy at the same frequency (extreme case: omni-directional noise, red curve) the SC gets flatter.

$|X_0(l, f)|$ is useful to become less dependent on the source signal: If $S(l, f)$ provides enough energy to suppress the influence of the noise terms $V_m(l, f)$, i.e. $S(l, f) \cdot H_m|_{\varphi_s}(l, f) \gg V_m(l, f)$, the source signal $S(l, f)$ in (4) cancels and only the ratio of the PDFRS remains. This ratio is known for a variety of source angles, because the PDFRS can be measured for a number Q of angles φ_q , $q = 0, \dots, Q - 1$. The basic reference feature vector $\mathcal{Y}(l, f, \varphi_q) = [\mathcal{Y}_1(l, f, \varphi_q), \mathcal{Y}_2(l, f, \varphi_q), \mathcal{Y}_3(l, f, \varphi_q)]^T$ is hence defined by

$$\mathcal{Y}_m(l, f, \varphi_q) = \frac{|H_m(l, f, \varphi_q)|}{|H_0(l, f, \varphi_q)|} \quad (5)$$

In case of multiple sources, reverberation or measurement noise the disturbance terms $V_m(l, f)$ in (4) cannot be neglected and the reference features in (5) are not appropriate. The difference between the cardioid channels decreases and hence the feature curves get compressed (cf. Fig. 3). With our planar array, the same thing happens for elevated sources (cf. (2)). To model all these effects, we extend our database with compressed versions of the clean features in (5).

$$\mathcal{Y}_m(l, f, \varphi_q, i) = \frac{|H_m(l, f, \varphi_q)| + G_i |\bar{H}_m(l, f)|}{|H_0(l, f, \varphi_q)| + G_i |\bar{H}_0(l, f)|} \quad (6)$$

where G_i , $i = 0, \dots, I - 1$ is a SNR-dependent weighting factor and $\bar{H}_m(l, f)$ is the mean of $H_m(l, f, \varphi)$ over φ . Compared to actually measuring features in noisy conditions, the advantage of the noisy reference feature model in (6) is that the measurement effort and memory requirements can be reduced significantly. By focusing only on a number of N_p strong, deterministic frequency components f_p , we can increase the performance in noisy environments, while reducing the computational complexity in the following processing steps. We obtain the peak-frequencies f_p by peak-picking in $|X_0(l, f)|$.

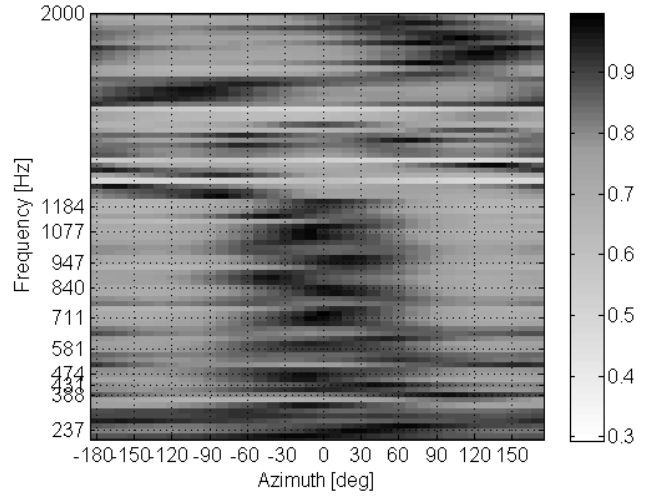


Figure 4: Similarity $C(f, \varphi_q)$. The source is a speech vowel signal located at $\varphi_s = 0^\circ$ mixed with omni-directional noise (6dB SNR). The peak-frequencies f_p are indicated on the ordinate.

2.4. Similarity Matching

We use a similarity measure based on the Euclidean norm:

$$\text{Sim}\{\mathbf{Y}, \mathcal{Y}\} = \frac{1}{1 + \sqrt{\sum_{m=1}^{M-1} |Y_m - \mathcal{Y}_m|^2}} \quad (7)$$

Eq. (7) yields values bound between 0 (completely dissimilar) and 1 (vectors are the same). The similarity between the l^{th} observed feature vector and the reference is computed for every reference position φ_q , peak frequency f_p and SNR index i .

$$C(l, f_p, \varphi_q, i) = \text{Sim}\{\mathbf{Y}(l, f_p), \mathcal{Y}(l, f_p, \varphi_q, i)\} \quad (8)$$

Instead of simply searching for the global maximum, we propose the following procedure: First, we compute an index $i_{\max}(l, f_p)$ that helps us to select the best matching SNR-version.

$$i_{\max}(l, f_p) = \underset{i}{\text{argmax}} \left\{ \max_{\varphi_q} \{C(l, f_p, \varphi_q, i)\} \right\} \quad (9)$$

Then we average over frequency which yields a single SC:

$$C(l, \varphi_q) = \text{mean}_{f_p} \{C(l, \varphi_q, f_p, i_{\max}(l, f_p))\} \quad (10)$$

To illustrate why we focus only on strong frequency components f_p , an example of the frequency-dependent SC is shown in Fig. 4. At the peak-frequencies f_p , the SC peaks close to the true source angle. At other frequencies, where the source does not provide sufficient energy, noise prevails and there is an increased likelihood of having a flat SC without a clear peak or a peak at a wrong angle.

2.5. Reliability Filtering

The azimuth estimate could be computed directly from $C(l, \varphi_q)$ as follows:

$$\hat{\varphi}_s(l) = \underset{\varphi_q}{\text{argmax}} \{C(l, \varphi_q)\} \quad (11)$$

If the frame does however contain mainly background noise (e.g. in a speaking pause), the estimate is likely to be different from the

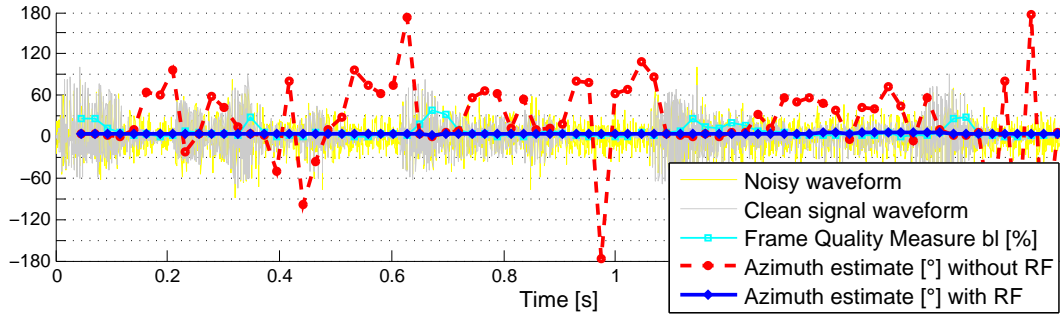


Figure 5: Effect of reliability filtering (RF) on a speech signal located at 0° with added omni-directional pink noise (SNR=0dB). By rating the quality of each frame between 0 (unreliable) and 1 (reliable) a stable, correct localization result can be obtained. Because the SNR is low, the frame estimate without RF (red) looks erratic and random. For the same reason, the frame quality measure b_l is close to zero most of the time. As can be seen from (12), the influence of such ‘bad frames’ is hence suppressed in the enhanced estimate (blue), i.e. we rely mainly on the previous estimate. Please note that at times where b_l (cyan) is significantly higher than 0, the frame-estimator (red) delivers the true result. This is the basis for a correct enhanced estimate (blue).

actual source direction. Hence, we rate the “frame quality” and smooth the run of $C(l, \varphi_q)$ over time l . We use a 1-pole lowpass-filter with a time-varying coefficient $0 \leq b_l \leq 1$:

$$\tilde{C}(l, \varphi_q) = b_l \cdot C(l, \varphi_q) + (1 - b_l) \cdot \tilde{C}(l-1, \varphi_q) \quad (12)$$

If $b_l = 0$ the SC is not updated, i.e. the previous SC is used. If $b_l = 1$ we rely only on the current frame and neglect the history. We compute b_l by rating the shape of the SC with a sample variance like metric:

$$\tilde{b}_l = \frac{1}{Q-1} \sum_{q=0}^{Q-1} (C(l, \varphi_q) - \text{mean}_{\varphi_q} \{C(l, \varphi_q)\})^2 \quad (13)$$

A flat SC achieves a low value whereas a SC with a clear, single peak achieves a high value of \tilde{b}_l . To ensure that b_l takes values close or equal to 1 under good conditions, we normalize and saturate \tilde{b}_l , i.e. $b_l = \max(\tilde{b}_l / \tilde{b}_{max})$, where \tilde{b}_{max} is obtained from a recording under perfect conditions (single source, free-field, high SNR). With $\tilde{C}(l, \varphi_q)$ in (12), the enhanced position estimate is given as

$$\hat{\varphi}_s(l) = \underset{\varphi_q}{\operatorname{argmax}} \{ \tilde{C}(l, \varphi_q) \} \quad (14)$$

Fig. 5 exemplifies the effect of reliability filtering (RF), i.e. the basic frame-level estimate in (11) is compared with the enhanced version in (14).

The azimuth estimate $\hat{\varphi}_s(l)$ in (14) is tied to the reference azimuth grid φ_q . To be able to produce results between the grid, interpolation between the maximum of the SC and its neighbors can be performed. We achieved good results with parabolic interpolation [10].

3. PRACTICAL EVALUATION

Recordings were carried out at the Institute of Electronic Music and Acoustics (IEM) in the “IEM-CUBE”, an approximately 10 x 12 x 4 m large room usually used as a lab, for lectures and electro-acoustic music (reverberation time $RT_{60} \approx 0.7$ s). The CUBE is equipped with an optical tracking system (OTS, a V624 data station and 15 M2 cameras by Vicon, cf. <http://www.vicon.com>) and a 24-channel hemispherical loudspeaker array (LSA).

As a sound source, we used 1) a loudspeaker and 2) a human speaker moving freely around the array. Both were tracked by the OTS for exact determination of the true source position (ground-truth) φ_l . The LSA was used for generation of omni-directional pink noise. To account for various SNRs, we added the appropriately weighted pink-noise recording to the clean target source recordings, i.e. the source recordings were made in quiet conditions (SNR between 25 and 45dB depending on the microphone, off/on-axis).

The reference database was obtained from impulse response (IR) measurements using a loudspeaker placed at different positions relative to the microphone array. To get smooth frequency responses and exclude noise and room reflections, these IRs were cut and windowed (to approx. 12ms) before transforming them to frequency domain. This makes the reference database more or less independent from the environment. We made experiments with a database recorded in a different room and achieved similar performance compared to a matched database.

The influence of diffuse reverberation is modeled via the noisy reference features in (6). It should however be noted that strong reflections from a dedicated direction may act as a competing source and can therefore impair the accuracy.

The placement of the array is not very critical because due to the normalization of the feature vector with the omni-directional channel, the characteristic pattern stays more or less the same. We compared placing the array on a desk with placement on the floor [10]. For the results shown in this paper, our array was placed on a small desk. We used a generic reference database (array placed on the floor, free-field) of our microphone with 10° azimuth resolution. If the azimuth resolution is coarser the accuracy may be impaired due to imperfect interpolation.

We compared our similarity approach (SIM) with 1), a time- and 2), a frequency-domain intensity vector (IV) localization approach (TDIV, and FDIV, respectively). We used 512 samples long, hamming windowed frames with 50 % overlap at a samplerate of 11025 Hz. We considered $N_p = 10$ peak frequency components between 200 and 4000 Hz for the SIM.

For the TDIV, the energy of each channel is computed in time domain and transformed to IV components [4]. We used a smoothing pole $a = 0.9$ to average the IV over time to achieve better results.

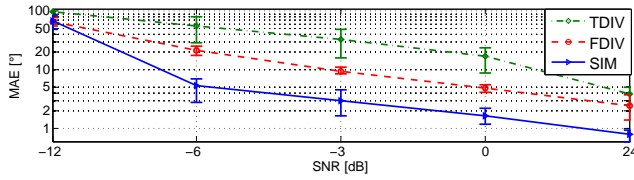


Figure 6: Static source position: Performance of different methods in terms of the mean (over φ_s) MAE. The errorbars indicate the first and the third quartile.

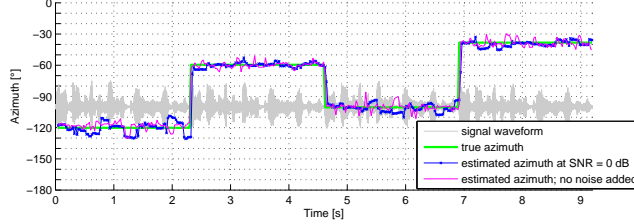


Figure 7: Source location jumps trough concatenation of different loudspeaker recordings (male speech). The estimate follows quickly.

The FDIV is based on a STFT and similar to the method described in [6]. Details on our implementation can be found in [10].

The estimation error is given as

$$\tilde{\varphi}_s(l) = \text{princarg} \{ \hat{\varphi}_s(l) - \varphi_s(l) \} \quad (15)$$

where the principle argument function can be defined using the modulo operator mod : $\text{princarg}(\varphi) = \text{mod} \{ \varphi + \pi, -2\pi \} + \pi$. As performance metrics, we used the mean absolute error MAE the root mean square error RMSE and the accuracy ACC_Δ , where $\delta_\Delta(\tilde{\varphi}_s) = 1$, if $|\tilde{\varphi}_s| \leq \Delta$ and 0 otherwise.

$$\text{MAE} = \text{mean} \{ |\tilde{\varphi}_s(l)| \} \quad (16)$$

$$\text{RMSE} = \sqrt{\text{mean} \{ \tilde{\varphi}_s(l)^2 \}} \quad (17)$$

$$\text{ACC}_\Delta = \frac{1}{L} \sum_{l=0}^{L-1} \delta_\Delta(\tilde{\varphi}_s(l)) \quad (18)$$

$\text{ACC}_5 = 90\%$ means for instance that 90% of all frames achieve an estimation error $|\tilde{\varphi}_s(l)| \leq 5^\circ$.

A short sentence (1.8s) of clean, male speech was played back from a loudspeaker positioned at 1m distance to the array. The elevation was 15° and the azimuth was varied between -180° and 0° in steps of 10° . A separate estimation result was computed for each angle. Fig. 6 shows the mean performance over all source angles in dependence of the SNR for the SIM, TDIV and FDIV. Our SIM-approach is very accurate and clearly superior to the TDIV and FDIV method. The exact values regarding the performance of our method are given in Table 1.

To assess the timing behavior of our algorithm, we concatenated the recordings from different azimuth angles, without pauses. Fig. 7 shows the true azimuth (optically tracked) and the estimate both for a recording with 0dB SNR and without added noise. Fig. 8 shows a similar plot, but for a male, human speaker walking around the array. More results can be found in [10].

SNR	-12	-6	-3	0	3	6	12	24
ACC_5	13	60	84.2	95.9	98.2	99.3	99.7	100
ACC_{10}	18.9	85.6	97.3	100	100	100	100	100
ACC_{15}	25.4	97.9	100	100	100	100	100	100
MAE	68.5	5.3	3.0	1.6	1.4	1.2	1.0	0.8
RMSE	84.9	6.0	3.4	2.1	1.8	1.4	1.3	1.0

Table 1: Performance of our SIM-approach with regard to static sources (Mean over $\varphi_s = (-180, -170, \dots, 0)^\circ$).

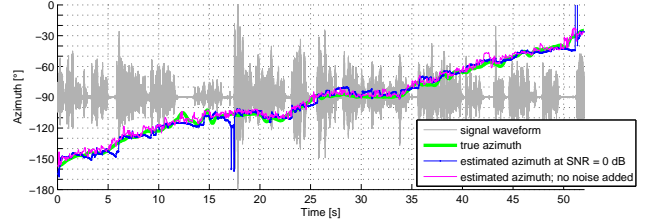


Figure 8: A male human speaks while walking around the array. The elevation was approx. $30^\circ < \vartheta_s < 40^\circ$, the radius $r_s \approx 1\text{m}$.

4. SUMMARY, CONCLUSION AND FUTURE WORK

We have presented a new method for tracking of a single sound source with a compact near-coincident microphone array (NCMA) that is cheap and handy. A reference feature database of the array has to be recorded once (free-field conditions, array placed on the floor). For source localization, a feature vector is computed frame-wise and compared to the database which yields a similarity curve (SC). We use a simple measure of the shape of the SC as a weight for the reliability of the current frame with respect to previous ones. With that, stable (no problems in signal pauses) and fast tracking can be achieved at the same time, without employing a separate detection algorithm. Conceptual advantages of our method are that we model the influence of noise and reverberation in our features and that we do not use fixed thresholds or time-constants. Practical experiments in a real room demonstrate the effectiveness of our approach, even in the presence of strong (-6 dB SNR) omni-directional noise.

The most obvious next working steps are a detailed study of the influence of microphone mismatch and evaluation of the performance in highly reverberant rooms. Our localization concept could be adapted to multi-source tracking and different array configurations, e.g. spherical arrays that also provide time-differences between the microphones. In contrast to our NCMA, such arrays allow for steering of higher order beam-patterns. Future work could also use the basic ideas behind our features and apply advanced pattern recognition approaches, e.g. a multi-class support vector machine.

5. REFERENCES

- [1] C. Zhang, D. Florencio, D.E. Ba, and Z. Zhang, "Maximum likelihood sound source localization and beamforming for directional microphone arrays in distributed meetings," *IEEE Transactions on Multimedia*, vol. 3, no. 3, pp. 538–548, 2008.

- [2] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*, Springer, Berlin, 2008.
- [3] J. DiBiase, H. Silverman, and M. Brandstein, “Robust localization in reverberant rooms,” in *Microphone Arrays: Signal Processing Techniques and Applications*, chapter 8, pp. 157–180. Springer, Berlin, 2001.
- [4] J. Merimaa and V. Pulkki, “Spatial impulse response rendering 1: Analysis and synthesis,” *J. Audio Eng. Soc.*, vol. 53, no. 12, pp. 1115–1127, December 2005.
- [5] B. Gunel, H. Hachabiboglu, and A.M. Kondo, “Acoustic source separation of convolutive mixtures based on intensity vector statistics,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 4, pp. 154–157, May 2008.
- [6] C. A. Dimoulas, K. A. Avdelidis, G. M. Kalliris, and G. V. Papanikolaou, “Improved localization of sound sources using multi-band processing of ambisonic components,” in *126th AES Convention, Munich*, May 2009.
- [7] O. Thiergart, R. Schultz-Amling, G. Del Galdo, D. Mahne, and F. Kuech, “Localization of sound sources in reverberant environments based on directional audio coding parameters,” in *127th AES Convention, New York*, Oct. 2009.
- [8] P. Cotterell, *On the Theory of the Second Order Soundfield Microphone*, Ph.D. thesis, University of Reading, 2002.
- [9] M. Holters, T. Corbach, and U. Zölzer, “Impulse response measurement techniques and their applicability in the real world,” in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09), Como, Italy*, September 1-4 2009.
- [10] K. Freiburger, “Development and evaluation of source localization algorithms for coincident microphone arrays,” M.S. thesis, Institute of Electronic Music and Acoustics, University of Music and Performing Arts, Graz, Austria, 2010.

A COMPARISON OF ANALYSIS AND RESYNTHESIS METHODS FOR DIRECTIONAL SEGMENTATION OF STEREO AUDIO

Jeremy J. Wells,

Audio Lab, Department of Electronics,
University of York, YO10 5DD
York, UK
jjw100@ohm.york.ac.uk

ABSTRACT

A comparison of analysis and resynthesis methods for use with a system for dividing time-coincident stereo audio signals into directional segments is presented. The purpose of such a system is to give greater flexibility in the presentation of spatial information when two-channel audio is reproduced. Example applications include up-mixing and transforming panning from amplitude to time-delay based. Included in the methods are the dual-tree complex wavelet transform and wavelet packet decomposition with best basis search. The directional segmentation system and the analysis and resynthesis methods are briefly described, with reference to the relevant underlying theory, figures of merit are presented for each method applied to three stereo mixtures of contrasting material and the subjective quality of the output (with links to all audio examples) is discussed.

1. INTRODUCTION

Audio recordings represent the capture of an acoustic event, or the rendering of an electronic/digital process, at a particular point in time. If there is more than one discrete channel then spatial information can be included in the recorded information. For two-channel stereo recordings, despite the sparsity of the spatial sampling points, a rich spatial experience can be provided for the headphone listener (particularly if the information is binaurally captured), or for (a) person(s) within a small listening area between two loudspeakers in a good listening environment. That said, there are now increased opportunities for surround sound (i.e. more than two-channel) storage, transmission and playback. Also, for individual listeners, the ideal presentation of the spatial information contained within a two-channel stereo audio recording will depend to a certain extent on their own preferences, listening environment and reproduction equipment. As trends in spatial presentation have varied over time, and continue to vary, so there may be a desire to revise the spatial presentation in existing two-channel recordings. Examples such as these require the ‘un-locking’ of the spatial information for each source (real and virtual) direction. This represents a considerable challenge where there are more source directions than channels.

The purpose of the target system, for which the analysis and resynthesis methods are compared here, is to divide the auditory scene presented by time-coincident (level-panned) audio into directional ‘segments’ [1]. Having more segments than audio channels offers flexibility in how each segment is presented at two (or more, if up-mixing is the application) loudspeakers. This is the over-arching aim of this research. As such, this work exists between individual source separation, such as that de-

scribed in [2], and spatial processing (for example [3-5]). The purpose is not necessarily to provide every single instrument separately for re-mixing, but to provide (distinct or overlapping) zones within a two-channel audio scene.

Previously an adaptive analysis/resynthesis method, based on dual-tree complex wavelets, was investigated and compared for use in this system with other methods traditionally used for this type of application [1]. Whilst the complex wavelet packets demonstrated an ability to adapt to the input, the figures of merit (FoM) used in that study demonstrated that they were always out-performed by another method (albeit not always the same one). However their adaptivity did avoid the transient smearing that was exhibited with short-time Fourier transform (STFT) methods with relatively long window lengths. A version of best basis search of complex wavelet packets, which used the available phase information was also investigated but did not consistently offer an improvement in the FoM and, in one case, caused a significant degradation in performance.

The work in this paper expands the range of analysis/synthesis methods used, introduces a regularised version of the phase-weighted best basis search and includes an additional FoM. Since subjective evaluation is also a crucial part of assessing these methods all audio examples used to generate the FoMs are discussed and made available online, as was done for the previous work.

In the next section of this paper an overview of directional segmentation of stereo audio is given and the segmentation system that all of the methods are tested with is described. Section 3 summarises the different analysis/resynthesis methods used and discusses the necessary theoretical detail. In section 4 the experimental design is explained and section 5 presents results for three different two-channel amplitude-panned mixtures. The final section summarises the paper and presents conclusions based on the results.

2. DIRECTIONAL SEGMENTATION OF TWO-CHANNEL AUDIO

2.1. Application examples

Space is represented in stereo recordings by differences between the signals reproduced at each loudspeaker (or earpiece if headphones are used, although only loudspeaker reproduction is considered in this paper). If there are no differences between the signals then the presentation is monophonic. The inter-channel differences may be amplitude (e.g. coincident microphones, typical panning controls), time (e.g. spaced microphones, time-delay

panning) and/or spectral (e.g. binaural with crosstalk cancellation for loudspeaker reproduction). A detailed discussion of the differences between amplitude- and time-difference presentation of audio via loudspeakers has been given previously [1, 6]. The work in those papers, and that presented here, is motivated by the desirability of reconfiguring spatial audio so that the spatial information can be presented in a different way. This work focuses on processing of amplitude panned (or captured) spatial audio.

If directional segments can be extracted from a two-channel mixture then they could be re-panned using time differences instead, therefore changing the presentation of spatial information. To introduce such position dependent delays for each source direction post-recording/mixing, where there are more source directions than channels, requires a separation system. The work described in this paper tests the effectiveness of different time-frequency analysis and synthesis methods when used in such a system.

Another means of changing the presentation is to change the number, or configuration, of loudspeakers. More than two-channels, delivered via the same number of loudspeakers (or more) can improve localisation, create a greater sense of envelopment and increase the size of the listening ‘sweet spot’. For soundfield reconstruction systems (such as high-order ambisonics) increasing the number of loudspeakers reduces spatial aliasing. For panning systems (e.g. so called ‘pair-wise’ positioning of sources) a greater number of discrete channels concentrates sound energy for a single source into a smaller number of speakers (or a smaller area of the array). This improves localisation over a wider listening area. For example, where there are more loudspeakers but just two discrete audio channels available (such as for the playback of legacy two-channel stereo over 5.1 surround systems) then the listening sweet spot may be enhanced (for example by extracting centre source directions and reproducing the audio via all of the front three speakers) or the spatial presentation may be enhanced by the positioning of source directions into rear speakers (e.g. for improved rendering of reverberation). This process is known as ‘up-mixing’ (e.g. [5]). Again, this process requires some form of separation algorithm in cases where there are more than two source directions.

2.2. Directional segmentation via time-frequency analysis and resynthesis

Time-frequency analysis, and resynthesis, is concerned with the decomposition, and construction, of signals as combinations of individual components that have certain positions and distributions in time and frequency [7]. The time-frequency plane for a signal is the distribution of these components across these two dimensions. An overview of the use of time-frequency analysis and resynthesis for directional segmentation of audio, along with a discussion of important prior work, is given in [1] and the reader is directed there for further information.

The context for the comparison of time-frequency analysis and resynthesis methods which is reported in this paper is a system that is described in detail in [1] and, again, the reader can find more information there. In that paper the possibility of using a phase-weighted entropy measure, in cases where the analysis-resynthesis method was both adaptive and complex, was examined. This phase-weighted entropy measure was given by:

$$H = - \sum_{p=1}^P \frac{(a_L(p) + a_R(p)) \log_2(a_L(p) + a_R(p))}{|\phi_R(p) - \phi_L(p)|} \quad (1)$$

where a and ϕ are the energy and phase of an individual atom of the decomposition, (L and R designate which spatial channel the atom belongs to) and H is the entropy for a particular basis of P atoms. It was found that this measure did not consistently improve performance. For this paper a regularised version of (1) is employed to investigate whether this improves consistency and/or performance, where r is the regularisation constant:

$$H = - \sum_{p=1}^P \frac{(a_L(p) + a_R(p)) \log_2(a_L(p) + a_R(p))}{|\phi_R(p) - \phi_L(p)| + r} \quad (2)$$

For real packet decompositions this version of the cost function cannot be used since no phase information is available. No best basis search is performed where the analysis-synthesis basis is fixed (i.e. the method is non-adaptive).

As described in [1] overlapping directional windows are used rather than the binary functions that have been commonly used in other studies (e.g. [2]). These directional windowing functions are shown in Figure 1 four equally spaced segments (which is the scenario tested in this paper). In most situations it will be desirable for a segment to be centred on a single source, and encompass that source only. In the case where sources are not regularly spaced, a modified windowing function would be required to ensure that segments are source-centred and preserve energy when combined. This could be achieved by using Hann-like tapering at the ends of constant functions as described in [26].

The windowing functions shown in Figure 1 only fully cover the front and rear quadrants (not the sides) of the recorded space. In anechoic situations where sources are only placed within the front quadrant (as is tested here) then the presence of energy outside of these regions (the residual after separation) indicates that separation has not been completely successful - the lower the energy level in the residual, the more successful the capture of sources within directional segments has been. Therefore the relative amount of energy in this residual is used as an FoM in the results presented in this paper. In echoic situations then this residual may also (correctly) contain reverberation/reflections from the side.

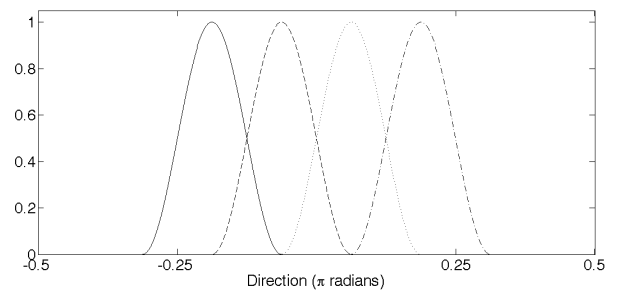


Figure 1: Directional segmentation windows applied to an audio scene containing four equidistantly and symmetrically spaced sources.

3. TIME-FREQUENCY ANALYSIS/SYNTHESIS METHODS

This section surveys the different analysis/resynthesis methods which are tested within the system discussed in sub-section 2.2. They can be grouped in two different ways: real and non-redundant versus complex and redundant, or adaptive versus non-adaptive. Since extensive coverage of many of the methods has been provided previously, what is presented here is a short summary of the information in [1], with additional detail on methods which have been used here for the first time.

3.1. Discrete wavelet transform (DWT)

This transform, which is exhaustively covered in the existing literature (e.g. [8]) is characterised by successive high and low pass filtering operations followed by decimation by a factor of two which yields a dyadic division of the time-frequency plane (fixed basis). The DWT is non-redundant, shift-variant and is sometimes referred to as the ‘fast’ or ‘decimated’ wavelet transform, to differentiate it from undecimated wavelet transforms (which are redundant). The nature of the wavelet (e.g. its distribution in time-frequency) is determined by the coefficients used in the filters. Four different sets of filter coefficients are used here. The first set are those of Daubechies with six vanishing moments (‘db6’, 12 tap filters), the second are Daubechies with fourteen vanishing moments (‘db14’, 28 tap) and the third are those of Vaidyanathan, designed for narrow transition from pass- to stop-band (‘vaid’, 24 tap) [8]. These filter sets are available either in the Mathworks Wavelet Toolbox, the Wavelab Toolbox or the Dual-Tree Wavelet Packet Toolbox [9-11]. The fourth set has been generated using the Filter Design Toolbox in Matlab (firpr2chfb function). These are 48 tap power-symmetric filters. The magnitude response of the low-pass filter is shown in Figure 2 (since the filters are power-symmetric the high-pass response is the exact reverse of that shown in the figure). In the experiments conducted for this paper, the DWT is carried out over eleven stages, yielding an eleven-scale decomposition. All four filter sets are orthogonal (i.e. the synthesis filters are the time reverse of the analysis filters).

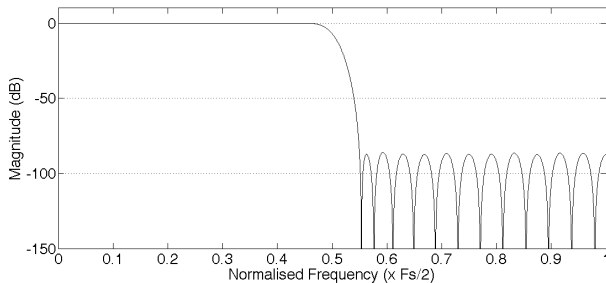


Figure 2: Magnitude frequency response of the 48 tap filter low-pass filter.

3.2. Wavelet packet decomposition (WPD)

The wavelet packet decomposition (WPD) is a generalisation of the DWT. Dyadic is just one of many different divisions of the time-frequency plane which are achieved when both low and high pass filtering operations are carried out on each set of coefficients at each decomposition level. A number of different decompositions can be achieved by different combinations of high-

and low-pass filtering operations and from these a single decomposition, offering a particular division of the time-frequency plane, can be chosen. Because of the binary tree structure of the decomposition, fast algorithms exist for searching for the best representation (the ‘best basis’) for a particular signal [12, 13].

The same four sets of filters that are used to implement the DWT are used for the WPD. Although the WPD can be considered to include the DWT, results for the DWT are presented separately in the next section since deriving a DWT only is a much cheaper operation computationally (but the basis is fixed). As for the DWT, the WPD is carried out over eleven scales, dividing the frequency axis into 2048 components for a full packet decomposition at this scale.

3.3. Cosine Packet Decomposition

Local cosine bases given by the Cosine Packet Decomposition (CPD) are also amenable to fast searching for a best basis [8]. The reader is directed to [1] for details of the implementation used in these experiments. The CPD divides the time-frequency plane into time partitions (whose frequency resolution are determined by choice of partition length), as opposed to the WPD, which divides the time-frequency plane into frequency partitions (whose length are determined by the choice of bandwidth) [8]. In both cases, many different combinations of different length (or bandwidth) segments can be chosen to form a number of orthogonal transforms (bases) from which a best basis can be chosen. As for the WPD with best basis, the CPD with best basis gives real coefficients of a non-redundant transform. The CPD is implemented here with the Wavelab toolbox [10]. A ‘sine’ taper is used and D is chosen so that the shortest packet is 512 samples long, given N . Where necessary the input signal is appended with zeros so that its length N is a power of two.

3.4. Short-time Fourier transform (STFT)

The STFT is perhaps the most widely known and well understood time-frequency analysis-resynthesis method for audio signals. A detailed discussion and description can be found in many sources (e.g. [14]). This method decomposes signals into equal length frames, which can be overlapping and tapered. A discrete Fourier transform (DFT) is applied to each frame and this gives a set of complex coefficients for sinusoids which are harmonics of the frame period, at the centre of each frame. The amount of overlap, and hence redundancy, can be arbitrarily set but is constrained by the shape of the tapering window applied to the frame (e.g. a minimum 50% overlap is required for the Hann window) and the distance from the centre of one frame to the next cannot be more than the frame length itself. Although the STFT can be non-redundant, tapering is usually applied to prevent energy spreading due to discontinuities at frame boundaries, and this renders the STFT redundant. For example, an overlap of 50% yields an STFT with 100% redundancy (providing zero-padding is not used). For the work described in this paper two sets of five STFT types are employed: one set applies a Hann window with 50% overlap prior to the DFT but no windowing of the output of the inverse DFT (IDFT), the second set has a 75% overlap and a Hann window is applied prior to DFT and after IDFT (where a Hann window is applied twice, the minimum overlap is 75%). Within each STFT set five frame lengths are used: 512, 1024, 2048, 4096 and 8192 samples. The frames are not zero-padded prior to analysis.

3.5. Dual-Tree Complex Wavelet Transform (DT-CWT)

The Dual-Tree Complex Wavelet Transform (DT-CWT) of Kingsbury is an extension of the DWT whereby a signal is decomposed by two sets of basis functions for which each corresponding pair of functions are approximately Hilbert transforms of each other [15]. As a result of this approach the DT-CWT is 100% redundant and approximately shift invariant. The Q-shift method of achieving approximate analyticity is used to determine the filter coefficients for level two of the decomposition onwards [16]. A different set of filter coefficients is used for the first stage of the transform: this filter set is used for both ‘trees’ with a one sample relative delay. At subsequent stages the Q-shift (quarter sample delay) filter set is used in both trees. In the second tree these filter coefficients are used in reverse order, giving a three-quarter delay and, therefore, the half sample relative delay between trees needed for analyticity. The longer the Q-shift filters are, the closer the two sets of basis functions are to being Hilbert transform pairs. Four sets of filter coefficients are used here to implement the DT-CWT: ‘db5’ (first stage) followed by the 14-tap Q-shift filter coefficients given in Table 2 of [15], ‘db14’ followed by the same 14-tap Q-shift filter coefficients, 24 tap Vaidyanathan followed by 24 tap Q-shift filters and, finally, the 48 tap filters described at the end of Section 3.1 followed by 48 tap Q-shift filters. The last two sets of Q-shift filters were designed using the Q-shift filter design toolbox [17]. For comparison the magnitude response of the 14, 24 and 48 tap Q-shift filters are shown in Figure 3. As for the real DWT, the number of scales in the following experiments is eleven.

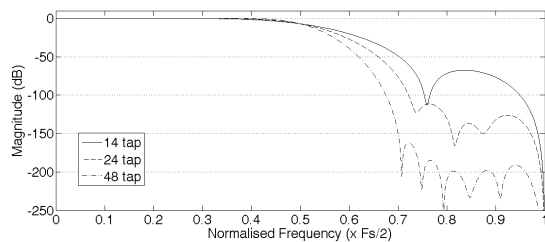


Figure 3: Magnitude frequency responses of each low-pass Q-shift filter.

3.6. Dual-Tree Complex Wavelet Packet (DT-CWPD)

The Dual-Tree Complex Wavelet Packet Decomposition (DT-CWPD) is the complex equivalent of the WPD, in the same way that the DT-CWT is the complex equivalent of the DWT. It yields bases with 100% redundancy. Since the DT-CWT consists of two orthogonal decompositions of the same signal, a straightforward approach to deriving a wavelet packet decomposition is to treat the two ‘trees’ as completely independent with their own sets of filters, where, after the first decomposition stage, the set used in one tree is the time-reverse of the set used in the second tree (as is the case for the DT-CWT). However ‘analyticity’ is better preserved by an altered scheme where some of the filtering stages of both trees use the same filters [18]. This scheme is employed here for the DT-WPD and it is implemented using (with some modifications) the toolbox provided at [19]. The same filter sets are used as for the DT-CWT (except that the first stage filter is ‘db5’ rather than ‘db6’, although it is replaced with ‘db6’ when non Q-shift filters are used in subsequent stages, see [19]). In fact, the first two filter sets are the same as those provided as examples at [19]. The maximum decomposition level is, again, eleven.

4. COMPARISON OF ANALYSIS/RESYNTHESIS METHODS

In order to compare the methods described in Section 3, they are tested using the system discussed at the end of Section 2. They are tested with three different anechoic audio mixtures, ranging from two to seven seconds in length, each containing four equally spaced point sources. The use of mixtures of anechoic sources allows the Signal to Residual Ratio (SRR, the ratio of energy in the residual segment to the energy contained in all of the other segments) to be used as an FoM. As for the experiments described in [1], for the purposes of this test the source positions for each mixture are the same and are known *a priori*. Whilst *a priori* knowledge of source positions is unlikely to be available in real-world applications it is the ability of the decomposition methods for segmentation which is specifically being tested here. In practice, *a posteriori* knowledge of source positions could be gained from global statistics for the mixture, such as the ‘panogram’ described in [5]. Each mixture contains four sources (src_{1-4}) and each of these are panned to the left and right outputs (out_L , out_R) of the mixture via:

$$\begin{pmatrix} \text{out}_L \\ \text{out}_R \end{pmatrix} = \begin{pmatrix} .8341, .5995, .4005, .1659 \\ .1659, .4005, .5995, .8341 \end{pmatrix} \begin{pmatrix} \text{src}_1 \\ \text{src}_2 \\ \text{src}_3 \\ \text{src}_4 \end{pmatrix} \quad (3)$$

This mixing matrix gives the same ratio between left and right energy that would occur for four sources spaced equidistantly in an arc within the front quadrant of a coincident pair of dipole microphones at 90 degrees to each other: sources positioned at -33.75 degrees ($-3\pi/16$ radians), -11.25 ($-\pi/16$), 11.25 ($\pi/16$) and 33.75 ($3\pi/16$) from the centre of the front quadrant. The centres of the windows shown in Figure 1 are at these positions and each position is covered by that one window only (at the centre of one window, the other three windows are at zero).

4.1. Mixture 1: pitched instruments

The individual sources for this mixture are clarinet, violin, soprano singer and viola performing an excerpt from a Mozart opera. The sources are obtained from [20].

4.2. Mixture 2: speech babble

This is a combination of four speakers talking simultaneously. The mixture comprises two male adults, one female adult and one male child. The sources are obtained from [21].

4.3. Mixture 3: percussion with single pitched instrument

This mixture consists of three hand percussion instruments and a single note with swept pitch from a Shakuhachi-like instrument. The sources are obtained from [22].

4.4. Figures of merit (FoM)

The quality of the segmentations is objectively measured by four quantities for each separated source: the energy weighted inter-channel correlation, the signal to residual energy ratio (SRR), the azimuth error and the signal to distortion ratio (SDR). The SDR is described in [23] and can be evaluated using the BSS_Eval Toolbox [24]. It compares the separated sources with the original

un-mixed sources and attempts to measure the ratio of the actual source energy to the energy due to artefacts of the separation algorithm and interference from other sources. It requires prior knowledge of the individual sources, which is available here. The SDR is designed for monophonic separated sources so it is applied here to the sum of each channel of the stereo separated outputs.

The other FoMs were introduced and used in [1] and so are only briefly summarised here. The zero-lag inter-channel cross-correlation between two channels for a single point source will be 1.0 since there are identical signals at each channel (albeit with different gains, if not positioned centrally) and there is no relative delay between them. Therefore, the closer this FoM is to 1.0, the better this segment has captured audio from one source only. The zero-lag cross correlation is given by:

$$X = \frac{\mathbf{src}'_L \cdot \mathbf{src}'_R}{\|\mathbf{src}'_L\| \|\mathbf{src}'_R\|} \quad (4)$$

where \mathbf{src}'_L and \mathbf{src}'_R are vectors containing the samples of the left and right channels of the segmented source. An overall FoM for all of the separated sources is given by the energy weighted mean of X of the sources. Whilst the SDR and the cross-correlation give an indication of the quality of the segmentation, the SDR does not take account of gain errors and the cross-correlation does not take account of gain or frequency response errors (it just measures the localisation of energy for a source – not how it is distributed in frequency). For anechoic sources the relative level of energy in the residual segment is an indicator of how successful the segmentation is in capturing the elements of the signal. The Signal to Residual ratio (SRR, measured in dB) is the ratio of the residual energy to the energy in the input mixture. The azimuths of individual separated sources can be calculated using

$$\theta' = \text{sgn}(|\mathbf{src}'_R| - |\mathbf{src}'_L|) \arccot \left(\frac{|\mathbf{src}'_R| + |\mathbf{src}'_L|}{|\mathbf{src}'_R| - |\mathbf{src}'_L|} \right) \quad (5)$$

and from this the azimuth error can be found, since actual the source directions are known. The energy-weighted mean azimuth error for all sources is an indicator of the extent to which segments are contaminated by each other, since azimuths will be biased by the presence of energy from other sources.

5. RESULTS

Three sets of plots are presented, one for each mixture. Within each set there are four plots which compare the performance of the different analysis and resynthesis methods for each FoM. The following abbreviations are used:

DT-CWPD 1, DT-CWT 1: 14 tap Q-shift filters, non Q-shift filters are 'db5' at the first stage, 'db6' thereafter.

DT-CWPD 2, DT-CWT 2: 14 tap Q-shift filters, non Q-shift filters are 'db14' at all stages.

DT-CWPD 3, DT-CWT 3: 24 tap Q-shift filters, non Q-shift filters are 24 tap Vaidyanathan filters.

DT-CWPD 4, DT-CWT 4: 48 tap Q-shift filters, non Q-shift filters are 48 tap Vaidyanathan filters.

WPD 1, DWT 1: 'db6' filters.

WPD 2, DWT 2: 'db14' filters.

WPD 3, DWT 3: Vaidyanathan 24 tap filters.

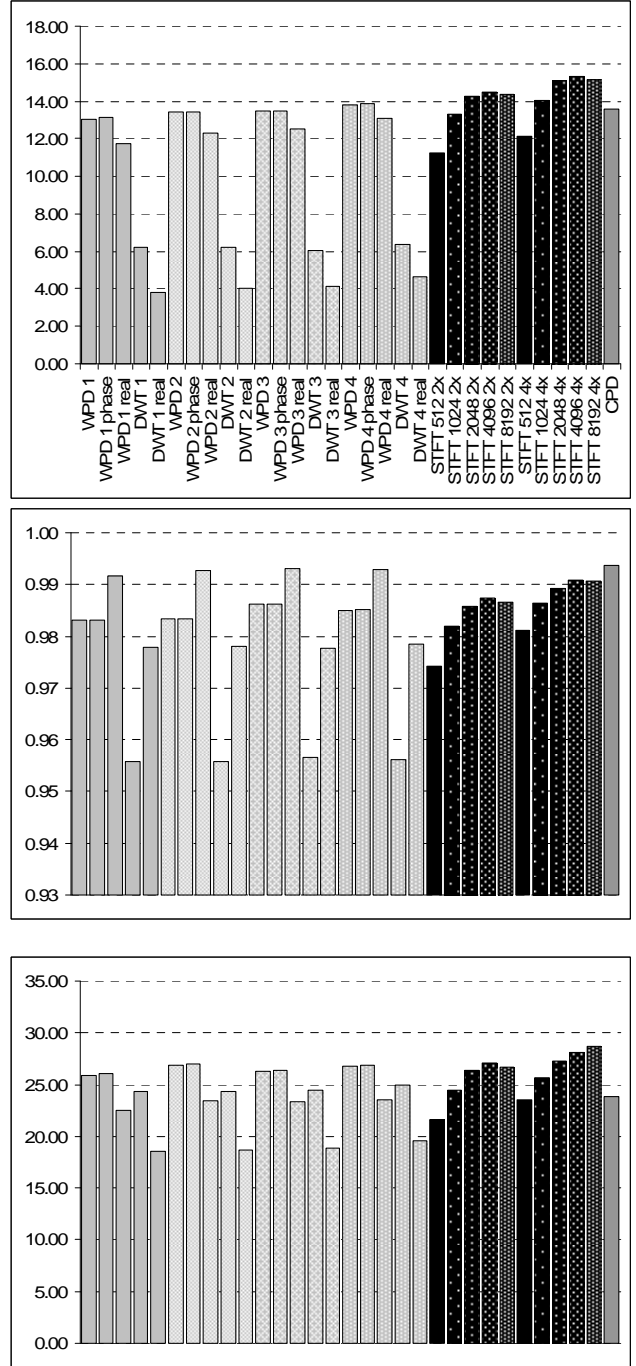
2x: STFT with 50% overlapping windows

4x: STFT with 75% overlapping windows

'Phase' indicates that the best basis has been determined using equation (2), rather than (1). The value of r , heuristically determined, is set at 0.01 for all mixtures.

For each set of figures, the x-axis labels, which indicate the type of analysis/synthesis method under test, are provided in the first of the four plots.

5.1. Figures of merit



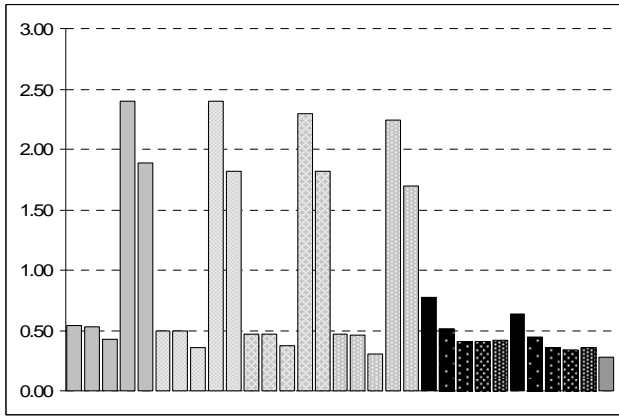


Figure 4: FoM for the instrument mixture: SDR (dB, top of previous page), correlation (middle of previous page), SRR (dB, bottom of previous page), azimuth error (radians, above)

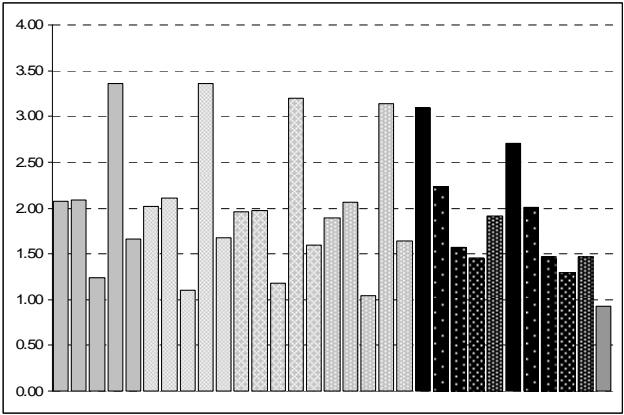
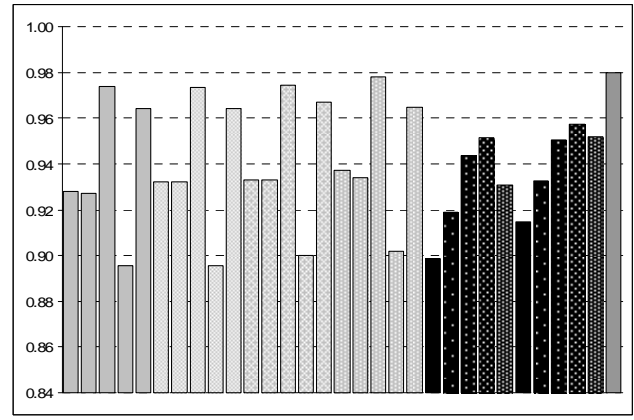
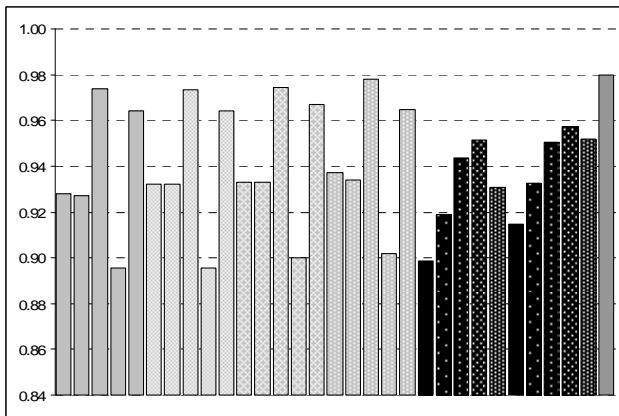
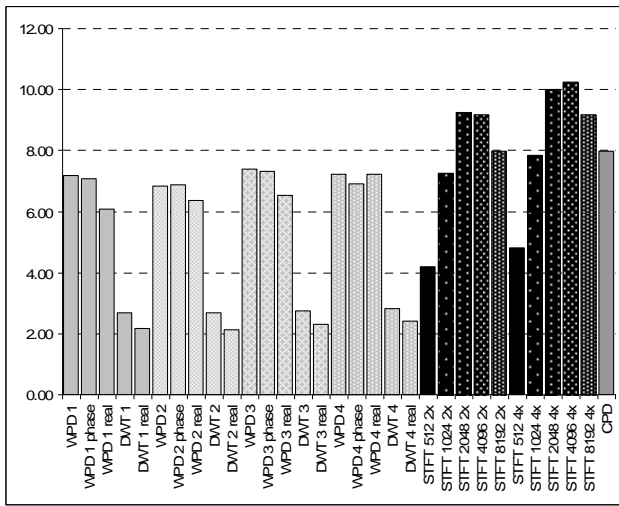
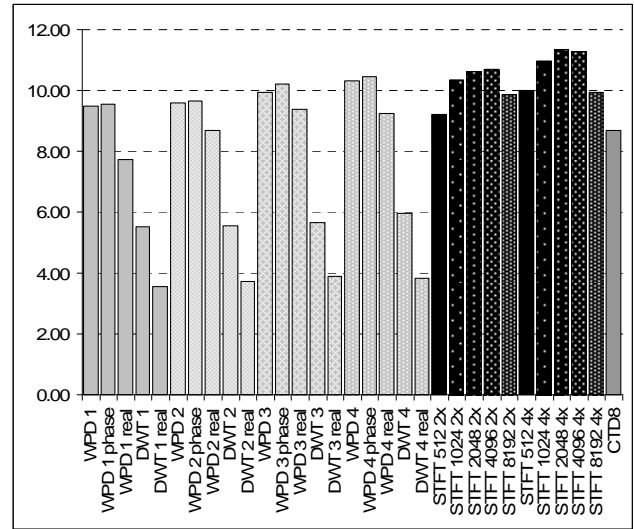


Figure 5: FoM for the speech mixture: SDR (dB), correlation, SRR (dB), azimuth error (radians)



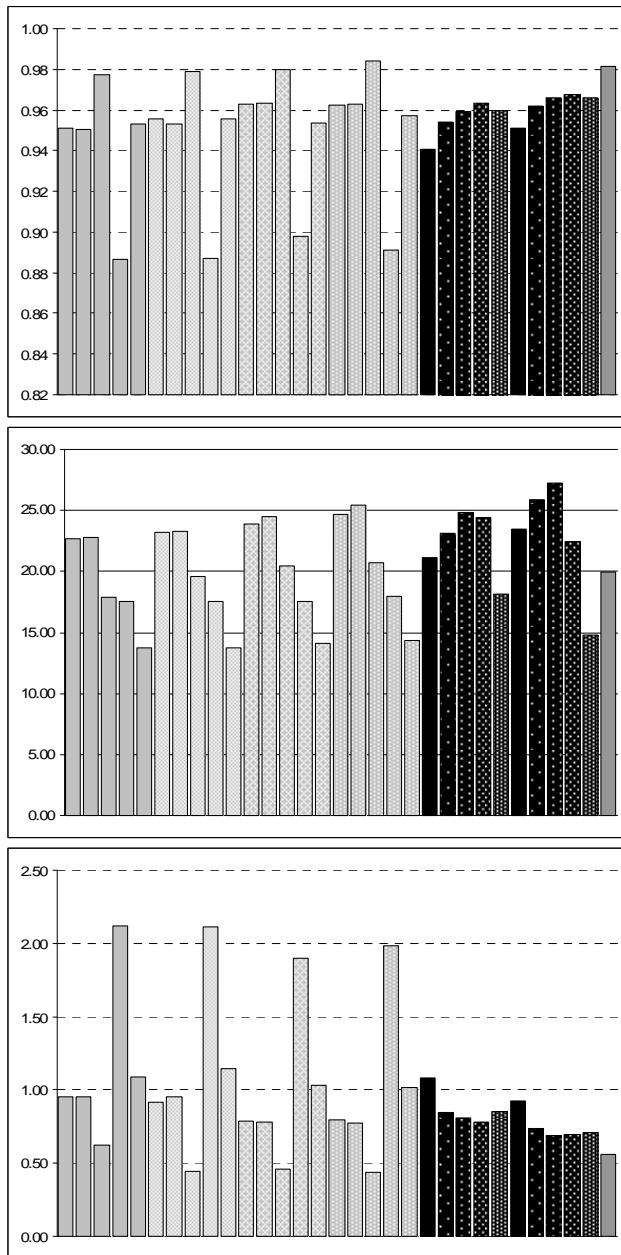


Figure 6: *FoM for the percussion mixture: SDR (top), correlation, SRR, azimuth error (bottom)*

5.2. Online audio examples

Audio files of the original sources, mixtures and separated sources for each method are provided in an online archive so that they can be auditioned [25].

5.3. Discussion

Some clear trends can be seen in Figures 4-6. Redundancy, whether achieved through introducing a second orthonormal transform whose basis functions are an approximate Hilbert transform pair with the first, or by increasing the overlap of basis functions improves the SDR performance of these methods for directional segmentation: STFTs using 75% overlapping win-

dows achieve better results than those using 50% overlap and the complex DWT or WPD always outperforms its real counterpart. Whilst ‘real’ methods do relatively well in terms of cross-channel correlation and azimuth error, they perform poorly in terms of SRR and their SDR performance is markedly worse than complex versions of the same methods in many cases. This shows that real analysis methods produce individual sources which have close to the correct azimuth and have narrow width, but this is at the cost of additional energy appearing in the residual.

The STFT with 75% overlap achieves the best FoMs for all mixtures. The 4096 frame-length STFT is best for the pitched instrument and speech mixtures, the 2048 frame-length version doing slightly better for the percussion mixture. The DT-CWPD using the fourth filter set performs best in terms of SDR out of the wavelet methods for all except the speech mixture. However it is out-performed by the CPD for all but the percussion mixture. As was found in [1], the use of phase-weighting in the entropy measurement for the best basis search does not have a dramatic positive impact on the FoMs. However the incorporation of a regularisation constant (not employed in [1]) does improve the consistency of phase-weighting overall (preventing serious anomalous degradations as occurred in [1]). Overall it is also more effective than the non-phase weighted measure, but the difference in performance is insufficient to be conclusive.

Listening to the audio outputs for the percussion mixture the drawback of long frame-length STFT analysis and resynthesis is clearly audible: transient smearing is much worse (although the separation is audibly better) than it is, for example, for the DT-CWPD with the filter set 3. The CPD performs well in the first half of the separation but then time definition is lost completely. Although transient smearing is both time-varying gain and spectral change, both of which the SDR should penalise, it does not have much impact on this FoM. It is worth noting here that in [2] the maximum STFT size was limited to 1024 because of the damage that longer frame sizes did to note onsets.

The longer-frame STFT methods audibly perform very well on speech and the pitched instrument mixture, although occasionally consonants and note onsets are degraded. Applying a window to the output of the IDFT, as well as the input to the DFT, is helpful in removing annoying ticks that are due to end-of-frame discontinuities introduced by the segmentation process.

6. CONCLUSIONS AND FUTURE WORK

This paper, along with its accompanying online resource of audio examples, has presented a comparison of a number of different time-frequency analysis/resynthesis methods for use in directional segmentation. The FoMs used clearly indicate that long-frame STFT methods with relatively high redundancy work best, although audition of the segmentations, particularly for percussion, provide a caution about using such objective measures as a sole indicator of quality. Whilst the dual-tree versions of the wavelet methods perform better than their real counterparts, and complex packets with long filters (including Q-shift) generally perform best, they do not begin to compete (numerically at least) with the STFT (or the CPD, considering just the speech mixture).

It is highly desirable to have an adaptive method that can perform as well as the STFT and there are many parameters and possibilities of the DT-CWPD that have yet to be fully investigated. Filters of 48 taps may still be too short for general audio

applications and the benefit of phase-weighting may become more apparent with longer Q-shift filters. The development of an adaptive method which can match the STFT's performance within the system, and on the example mixtures, tested here, remains a challenge. However the challenge is a worthwhile one, given the potential benefits of high-quality directional segmentation. Of course, some consideration should also be given to computational cost, and more redundant methods are usually more expensive. But, for this application, redundant time-frequency representations seem to perform best overall.

7. REFERENCES

- [1] J. Wells, "Directional Segmentation of Stereo Audio via Best Basis Search of Complex Wavelet Packets", presented at the 130th Audio Engineering Society International Convention, London, UK, 2011 Convention, Preprint No. 8436.
- [2] A. Nesbit et al., "Audio Source Separation with a Signal-Adaptive Local Cosine Transform", *Signal Processing*, Vol. 87, No. 8, August 2007, pp. 1848-1858.
- [3] V. Pulkki, "Spatial Sound Reproduction with Directional Audio Coding", *Journal of the Audio Engineering Society*, Vol. 55, No. 6, June 2007, pp. 503-516.
- [4] C. Faller, "Modifying the Directional Responses of a Coincident Pair of Microphones by Postprocessing", *Journal of the Audio Engineering Society*, Vol. 56, No. 10, October 2008, pp. 810-822.
- [5] C. Avendano and J. Jot, "A Frequency Domain Approach to Multichannel Upmix", *Journal of the Audio Engineering Society*, vol. 52, No. 7/8, July/August 2004, pp. 743-749.
- [6] J. Wells, "Modification of Spatial Information in Coincident Pair Recordings", presented at the 128th Audio Engineering Society International Convention, London, UK, 2010, Preprint No. 7983.
- [7] L. Cohen, *Time-Frequency Analysis*, Prentice Hall, 1994.
- [8] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd edition, Academic Press, 1999.
- [9] M. Misiti et al., "Wavelet Toolbox User's Guide", available from <http://www.mathworks.com>
- [10] Donoho, D. et al, "The WaveLab Matlab Toolbox", available from: <http://www-stat.stanford.edu/~wavelab/>
- [11] Bayram, I., "The Dual-Tree Complex Wavelet Packet Transform Matlab Toolbox", available from <http://web.itu.edu.tr/~ibayram/dtcwpt/>
- [12] M. Wickerhauser, *Adapted Wavelet Analysis from Theory to Software*, A K Peters, 1994.
- [13] R. Coifman and M. Wickerhauser, "Entropy Based Algorithms for Best Basis Selection", *IEEE Transactions on Information Theory*, Vol. 38, No.2, pp. 713-718, 1992.
- [14] U. Zölzer, Ed., *DAFX – Digital Audio Effects*, J. Wiley & Sons, 2002.
- [15] I. Selesnick et al, "The Dual-Tree Complex Wavelet Transform", *IEEE Signal Processing Magazine*, pp. 123-151, 2005.
- [16] N. Kingsbury, "Complex Wavelets for Shift Invariant Analysis and Filtering of Signals", *Journal of Applied and Computational Harmonic Analysis*, Vol. 10, No. 3, pp. 234-253, 2001.
- [17] N. Kingsbury, "Complex Wavelet Design Package", available from <http://www-sigproc.eng.cam.ac.uk/~ngk/>
- [18] I. Bayram and I. Selesnick, "On the Dual-Tree Complex Wavelet Packet and M-Band Transforms", *IEEE Transactions on Signal Processing*, Vol. 56, No. 6, pp. 2298-2310, 2008.
- [19] I. Bayram, "The Dual-Tree Complex Wavelet Packet Transform Matlab Toolbox", available from <http://web.itu.edu.tr/~ibayram/dtcwpt/>
- [20] T. Lokki et al., "Anechoic Recordings of Symphonic Music", available at <http://auralization.tkk.fi/>
- [21] Howard, D. et al., CD of audio examples accompanying Howard, D. and Angus, J., *Acoustics and Psychoacoustics* (3rd Edition), Focal Press, London, 2006.
- [22] Various, *Roland Sampling Showcase*, Time and Space Audio CD, 1994.
- [23] E. Vincent et al., "Performance measurement in blind audio source separation", *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 14, No. 4, pp. 1462 – 1469, 2006.
- [24] C. Févotte et al., "BSS_Eval Toolbox User Guide Revision 2.0", available at http://bass-db.gforge.inria.fr/bss_eval/user_guide.pdf
- [25] Audio examples and Matlab code for this paper is available at: Audio examples for this paper available at: www.jezwells.org/directional_segmentation.
- [26] A. Master, "Stereo Music Source Separation via Bayesian Modeling", PhD Thesis, Stanford University, USA, 2006.

FAUST-STK: A SET OF LINEAR AND NONLINEAR PHYSICAL MODELS FOR THE FAUST PROGRAMMING LANGUAGE

Romain Michon,*

CIEREC, EA 3068
Université Jean Monnet
F-42023, Saint-Etienne, France
rmichon@ccrma.stanford.edu

Julius O. Smith,

Center for Computer Research in Music and Acoustics.
(CCRMA) Stanford University
Palo Alto, CA 94305, USA
jos@ccrma.stanford.edu

ABSTRACT

The *FAUST Synthesis ToolKit* is a set of virtual musical instruments written in the FAUST programming language and based on waveguide algorithms and on modal synthesis. Most of them were inspired by instruments implemented in the *Synthesis ToolKit* (STK) and the program *SynthBuilder*.

Our attention has partly been focused on the pedagogical aspect of the implemented objects. Indeed, we tried to make the FAUST code of each object as optimized and as expressive as possible.

Some of the instruments in the FAUST-STK use nonlinear all-pass filters to create interesting and new behaviors. Also, a few of them were modified in order to use gesture data to control the performance. A demonstration of this kind of use is done in the *Pure Data* program.

Finally, the results of some performance tests of the generated C++ code are presented.

1. INTRODUCTION

The *FAUST Synthesis ToolKit*¹ is set of virtual musical instruments programmed in the FAUST² programming language. Most of them are based on physical models inspired from the algorithms implemented in the *Synthesis ToolKit* (STK)³ [1] and the program *SynthBuilder* [2].

The STK has been developed since 1996 by P. R. Cook and G. P. Scavone. It is a set of open source audio signal processing and algorithmic synthesis classes written in the C++ programming language that can be used in the development of music synthesis and audio processing software.

SynthBuilder was a program developed at Stanford's CCRMA⁴ in the nineties to implement sound synthesis based on physical models of musical instruments. Most of its algorithms use the waveguide synthesis technique but some of them are also based on modal synthesis [3].

An important part of our work consisted of improving and simplifying the models from these two sources in order to make them

more efficient thanks to the FAUST semantic. All FAUST code in the FAUST-STK is commented, including frequent references to external bibliographical elements. Finally, many of the algorithms from the STK and *SynthBuilder* were upgraded with nonlinear all-pass filters [4].

First, we discuss the different models of musical instruments implemented in the FAUST-STK, noting problems encountered and how they were resolved. Finally, we'll present performance measurements for the generated C++ code.

2. WAVEGUIDE MODELS

Waveguide synthesis of string and wind instruments was introduced during the 1980s [5, 6, 3]. It can be viewed as a generalization of either the Kelley-Lochbaum vocal-tract model [7, 8] or Karplus-Strong "digital" algorithm [9, 3]. Waveguide synthesis makes it possible to model any kind of string, bore, or vibrating structures with a network of delay lines and filters. Waveguide instruments are very suitable for implementation in the FAUST language because of their 1D "stream like" architecture.

We now give a brief overview of the FAUST-STK waveguide instruments.

2.1. Wind Instruments

The algorithms used in the FAUST-STK are almost all based on instruments implemented in the *Synthesis ToolKit* and the program *SynthBuilder*. However, it is important to observe that some of them were slightly modified in order to adapt them to the FAUST semantic.

An attempt was made to use functions already defined in the default FAUST libraries to build our models. However, new support functions were written as needed in order to be able to use parameters from the STK classes and the *SynthBuilder* patches verbatim, without transformation or embedding within more general functions. The added functions were placed in a file called `faust-stk/instrument.lib`.

All the wind instruments implemented in the FAUST-STK are based on a similar architecture. Indeed, in most cases, the breath pressure that corresponds to the amplitude of the excitation is controlled by an envelope. The excitation is used to feed one or several waveguides that implement the body of the instrument. For example, in the case of a clarinet, the excitation corresponds to the reed that vibrates in the mouthpiece, and the body of the instrument is the bore and the bell. In Figure 1, it is possible to see the block diagram of one of the two clarinet models that are implemented in the

* CCRMA visiting researcher from Saint Étienne University, France. Work carried out in the frame of the ASTREE Project (ANR-08-CORD-003).

¹`<faust-distribution>/examples/faust-stk/`

²Functional Audio Stream is programming language that proposes an abstract, purely functional approach to signal processing. It has been developed at Lyon's GRAME (Groupe de recherche en Acoustique et en Musique Electronique) since 2002: <http://faust.grame.fr/>.

³<https://ccrma.stanford.edu/software/stk/>

⁴Center for Computer Research in Music and Acoustics

FAUST-STK. In that case, an ADSR⁵ envelope that is embedded in the *breathPressure* box controls the breath pressure.

The other clarinet implemented in the FAUST-STK is a bit more complex as it has a tone hole model that makes it possible to change the pitch of the note being played in a more natural way. Indeed, in the algorithm shown in Figure 1 and as in most of the basic waveguide models, the pitch is modulated by changing the length of the loop delay line which would correspond in “the real world” to changing dynamically the size of the clarinet’s bore during the performance, as if it were a trombone.

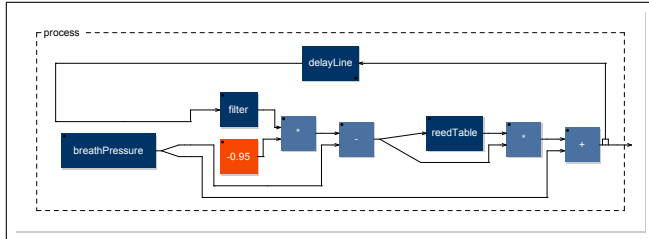


Figure 1: *clarinet.dsp* algorithm drawn by FAUST using *faust2svg*.

The reed table employed with the two clarinets to excite the model was also used to create a very simple saxophone model that is even more comparable to a violin whose strings are excited by a reed.

Two models of flute are implemented in the FAUST-STK. The first one is based on the algorithm used in the *Synthesis ToolKit* that is a simplified version of [10]. The other model is showed in Figure 2. It uses two loops instead of one.

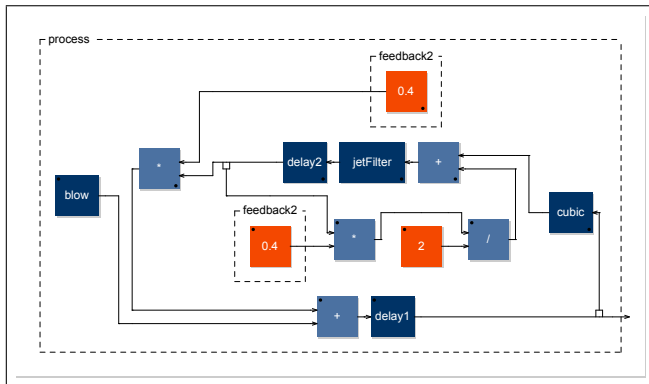


Figure 2: *flute.dsp* algorithm drawn by FAUST using *faust2svg*.

A simple model of a brass instrument inspired from a class of the *Synthesis ToolKit* and with a mouthpiece based on the model described in [11] is implemented in the FAUST-STK. It can be used to emulate a wide range of instrument such as a french horn, a trumpet or even a trombone. Its algorithm can be seen in Figure 3.

⁵Attack - Decay - Sustain - Release.

Finally, a tuned bottle in which it is possible to blow through the neck to make sound is also implemented in the FAUST-STK.

2.2. String Instruments

Some waveguide synthesis algorithms for plucked strings have previously been implemented in FAUST [12], and elements of these ports appear in the libraries *filter.lib* and *effect.lib* within the FAUST distribution. Going beyond these, the FAUST-STK includes models of stringed instruments from the STK such as a Sitar, bowed-string instrument, and *SynthBuilder* patches (running on a NeXT Computer) for an acoustic bass, piano, and harp-sichord. Most of these models were furthermore extended with the new nonlinear allpass for spectral enrichment [4]. Further discussion regarding the nonlinear allpass and synthesis of keyboard instruments is given below in §3 and §6, respectively.

2.3. Percussion Instruments

Four objects in the FAUST-STK use the banded waveguide synthesis technique (described in [13]) to model the following percussion instruments:

- an iron plaque;
- a wooden plaque;
- a glass harmonica;
- a tibetan bowl.

Each of them can be excited with a bow or a hammer.

3. USING NONLINEAR PASSIVE ALLPASS FILTER WITH WAVEGUIDE MODELS

Some of the instruments implemented in the FAUST-STK are using nonlinear passive allpass filters in order to generate nice natural and unnatural sound effects [4]. Nonlinear allpass filters can add interesting timbral evolution when inserted in waveguide synthesis/effects algorithms. The nonlinearities are generated by dynamically modulating the filter coefficients at every sample by some function of the input signal. For the instruments that use this kind of filter in the FAUST-STK, the user can decide whether the coefficients are modulated by the input signal or by a sine wave. In both cases, a “nonlinearity factor” parameter scales the range of the modulation of the filter coefficients. This parameter can be controlled by an envelope in order to make the modulated behavior more natural.

We adjust the length of the delay line of the instruments that use nonlinear allpass filters in function of the nonlinearity factor and of the order of the filter as follows:

$$DL = (SR/F) - FO \times NF \quad (1)$$

where DL is the delay length in samples, SR is the sampling rate, F is the pitch frequency, FO is the filter order and NF the nonlinearity factor (value between 0 and 1).

The nonlinear allpass filter can be placed anywhere in the waveguide loop, for example just before the feedback as showed in Figure 4.

Finally, it is interesting to mention that we were able to implement a frequency modulation synthesizer in the FAUST-STK by using this kind of filter on a sine wave signal. A related result is reported in [14].

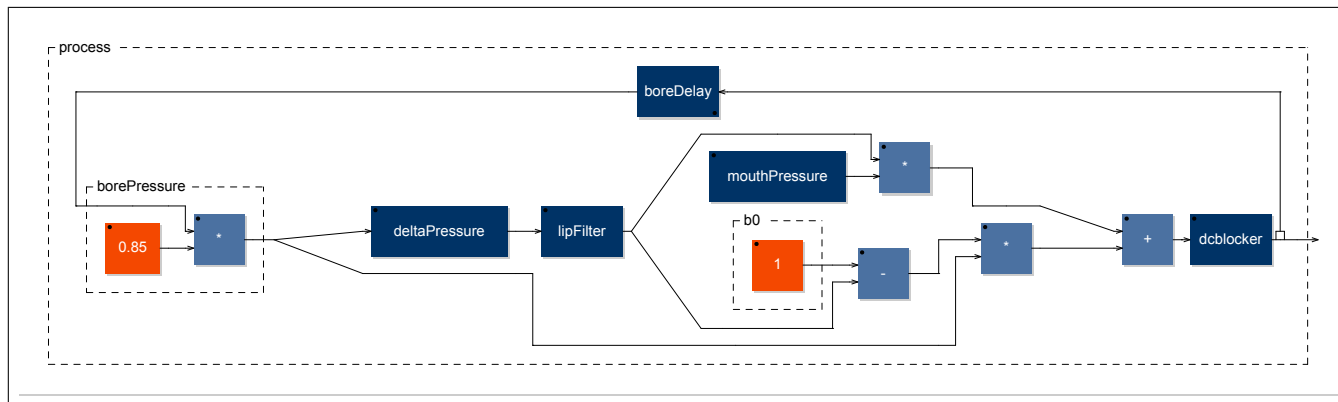


Figure 3: *brass.dsp* algorithm drawn by FAUST using *faust2svg*.

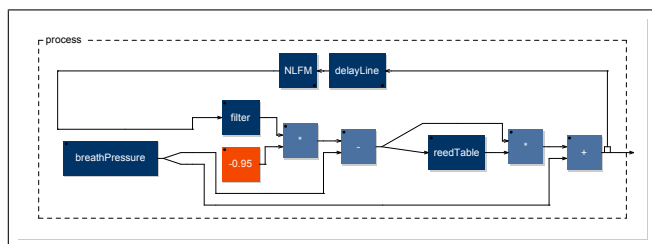


Figure 4: *Modified version of `clarinet.dsp` (Cf. figure 1) that uses a nonlinear allpass filter in its feedback loop.*

4. MODAL MODELS

A set of instruments using modal synthesis can be found in the FAUST-STK. They are all implemented in the same code as they are based on the same algorithm.

Modal synthesis was developed primarily by J-M. Adrien in the 1980s [15] and is very similar to the banded-waveguide technique (§2.3), as it consists of exciting a filter-bank with an impulse (Figure 5). The source-filter approach to sound synthesis has long been used for voice synthesis [16].

Implementing modal synthesis with FAUST was a bit challenging, as it requires handling a large number of parameters and an excitation signal stored in a wave file. The first problem was solved by using the foreign-function primitive in FAUST which allows using a C++ function within FAUST code. The different values were stored in an array of floats used in a function that takes an index as an argument and that returns the corresponding number.

To solve the other problem of importing a wavetable from a sound file in a FAUST object, we first tried to use the `libsndfile` library developed by E. de Castro Lopo [17] that makes it possible to easily handle wave files in C++. Unfortunately, it appears that this solution was not compatible with all the FAUST architectures. Based on this observation and the fact that the wave tables used in the STK had a maximal size of 1024 samples, we decided to use the same technique as the one previously explained. Indeed, the raw data were extracted from the wave file to be put in an array of floats that can be used in a C++ function to return the values with

an index. This C++ function can then be called in FAUST using the foreign-function mechanism to fill a buffer with the `rdtable` primitive.

5. VOICE SYNTHESIS

A very simple voice synthesizer based on the algorithm from the *Synthesis ToolKit* is implemented in the FAUST-STK. It uses a lowpass-filtered impulse-train to excite a bank of 4 bandpass filters that shape the voice formants. The formant parameters are stored in a C++ function in the same way described in §4 as a set of center frequencies, amplitudes, and bandwidths. This function is then called in the FAUST code using the foreign function primitive. The thirty-two phonemes stored in this function are the same as in the *Synthesis ToolKit*.

6. KEYBOARDS

A *SynthBuilder* patch implementing a commuted piano [18] was written in the late 1990s at Stanford’s CCRMA. This patch was partly ported in 2006 by Stephen Sinclair at McGill University in the *Synthesis Toolkit* [19]. A big part of his work consisted of extracting parameter-values from the *SynthBuilder* patch and storing them in a set of C++ functions. We reused them to build our FAUST commuted piano version by using the *foreign function* mechanism as described in §4.

In this piano model, the keyboard is split into two parts, each using a different algorithm: The tones below $E6$ use the commuted waveguide synthesis technique [3] while tones above or equal to $E6$ use modal synthesis (a series of biquad filters) to generate the sound (Figure 6).

A commuted harpsichord has also been implemented in the FAUST-STK. It was inspired by another *SynthBuilder* patch that uses a very similar algorithm to the one described above.

The current FAUST versions of the commuted piano and harpsichord are not polyphonic. However, the *faust2pd* program developed by Albert Gräf [20] makes it possible to automatically produce *Pure Data* patches that implement polyphonic synthesizers that use FAUST generated Pd plug-ins. They can then be controlled via MIDI or OSC directly in Pure Data.

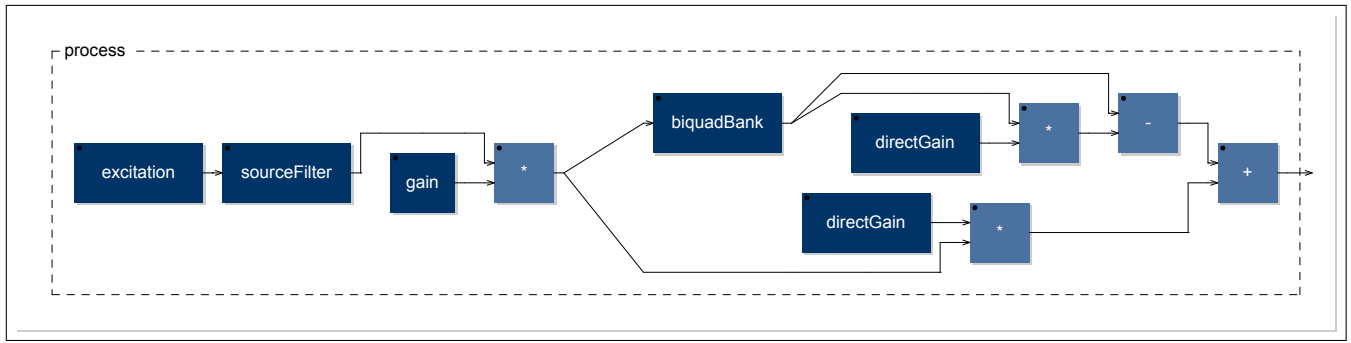


Figure 5: *modalBar.dsp* algorithm drawn by FAUST using *faust2svg*.

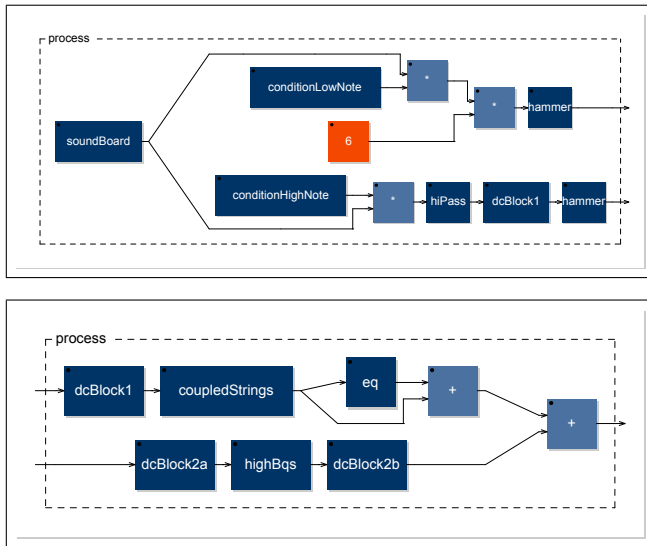


Figure 6: *Commuted piano* algorithm drawn by FAUST using *faust2svg*. The upper figure is the beginning of the model and the lower figure the end.

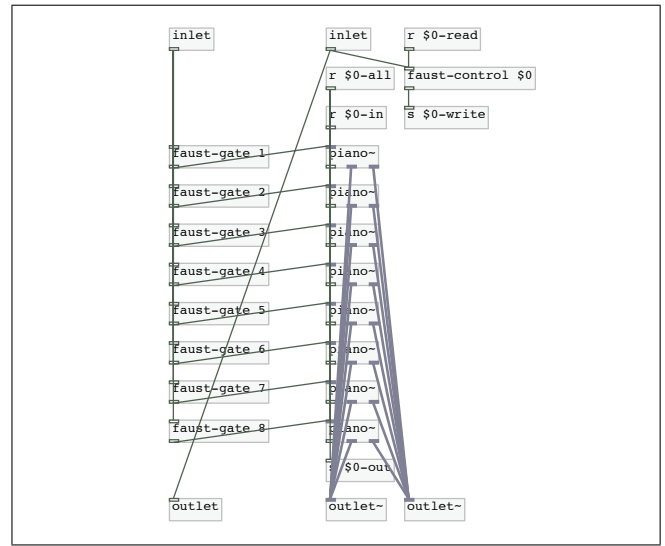


Figure 7: Synthesis part of the *Pure Data* polyphonic sub-patch generated with *faust2pd* from "*piano.dsp*". In the current case, a height voices polyphony synthesizer is implemented so *piano~.dsp* is called *height times*.

7. USING A FAUST-STK PHYSICAL MODEL WITH GESTURE-FOLLOWING DATA

Parameter values are very important when dealing with physical modeling. Indeed, even if in most cases it is possible to produce nice sounds with static values for each parameter, the sound quality can be improved a lot by using dynamic values that can describe better the state of the model as a function of the note and the amplitude being played.

E. Maestre worked during his PhD on modeling the instrumental gesture for the violin [21] at the MTG.⁶ With his help, it was possible to modify the algorithm of the bowed instrument from the *STK* in order to make it compatible with gesture data. The following changes were performed on the model:

- the *ADSR* used to control the bow velocity was removed;

⁶Music Technology Group, University Pompeu Fabra, Barcelona (Spain).

- a "force" parameter that controls the slope of the bow table was added;
- a switch was added at the output of the bow table;
- we created a four-string violin where it is possible to modify the value of the parameters of each string independently;
- the simple body filter was replaced by a bank of biquad filters that impart a violin body response on the generated sound;
- an improved reflection filter also based on a bank of biquads is used.

The FAUST code was used to create a *Pure Data* plug-in. The gesture data for each physical parameter (note frequencies, bow position, bow velocity, bow force, and number of the string to be used) of the violin model were placed in separated text files that can be used in a *Pd* patch. In the example shown in Figure 8, the

values are changed every 4.167 milliseconds. The gesture dataset used plays a traditional Spanish song called Muiñeira.

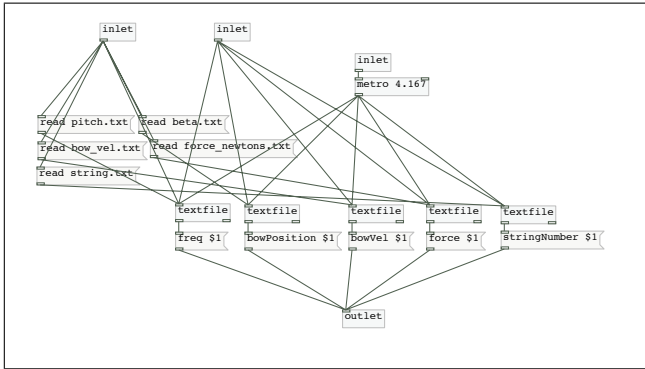


Figure 8: Pure Data sub-patch used to send the gesture data for Muiñeira in the FAUST generated plug-in.

8. OPTIMIZATION AND PERFORMANCE

8.1. File size

Digital signal processing algorithms can be expressed very compactly in FAUST. The reduction in code size over C++ or even matlab implementations is most of the time very significant. Thereby, we tried to make the FAUST-STK algorithms as concise and readable as possible.

It is difficult to compare the STK C++ and FAUST source, because most of the physical models in the *Synthesis ToolKit* were implemented using several functions spread-out among different files. Moreover, these functions may contain information not related to the algorithm itself.

We nevertheless carried out a source-size comparison as a rough guide, and the results are given in Table 1 (last page). We took into account, in both FAUST and C++, the implementation of the algorithm itself, and the code concerning parameter-handling. While the precision of the comparison is open to debate, we see clearly that the FAUST code is generally more compact than the C++.

8.2. CPU load

The FAUST compiler optimizes the efficiency of its generated C++ code. Thus, we tried to compare for some models the CPU load between Pure Data plug-ins created using the *stk2pd*⁷ program with Pd plug-ins generated by FAUST using the Pure Data architecture file.

In both cases, Pd plug-ins were compiled in 32 bits and the signal processing is scalar. Tests were carried out on a MacBook Pro with the following configuration:

- processor: 2.2 GHz Intel Core 2 Duo;
- RAM: 2GBytes DDR2.

Results of this comparison can be seen in Table 2.

⁷*stk2pd* is a program that was developed at Stanford’s CCRMA by M. Gurevich and C. Chafe. It converts any C++ code from the *STK* into a plug-in for Pure Data [22].

FAUST file name	STK	FAUST	Difference
blowBottle.dsp	3.23	2.49	22.91
blowHole.dsp	2.70	1.75	35.19
bowed.dsp	2.78	2.28	17.99
brass.dsp	10.15	2.01	80.20
clarinet.dsp	2.26	1.19	47.35
flutestk.dsp	2.16	1.13	47.69
saxophony.dsp	2.38	1.47	38.24
sitar.dsp	1.59	1.11	30.19
tibetanBowl.dsp	5.74	2.87	50

Table 2: Comparison of the performance of Pure Data plug-ins using the STK C++ code with their FAUST generated equivalent. Values in the “STK” and “FAUST” columns are CPU loads in percents. The “difference” column give the gain of efficiency in percents.

As the original STK C++ code is already very well written and optimized, this comparison shows how efficient the FAUST compiler is at generating highly optimized C++ codes.

9. CONCLUSIONS

Even if the primary goal of the FAUST-STK is the use of its physical models in a musical manner, it was also built to be a pedagogical tool. Indeed, because of its transparency and efficiency, the FAUST programming language is particularly suitable for teaching digital audio signal processing. Therefore, a clean and well commented FAUST program is arguably the best way to document the implemented instruments, especially in view of the automatic block-diagram facility. FAUST also has the advantage of being committed to a stable computational specification, unlike C++ in which the meanings of “long” and “short” may change over time, for example, or even across computing platforms.

With its continually growing user community, FAUST is becoming a high quality tool for the implementation of audio digital signal processing algorithms. The number of filters, effects and sound synthesizers available in FAUST is constantly increasing. The combined forces of JACK⁸ and FAUST recently upgraded by the possibility to control the generated programs with the OSC⁹ communication standard constitute a high efficiency work platform whose limits are only constrained by one’s imagination.

10. REFERENCES

- [1] P. Cook, “The Synthesis ToolKit (STK),” in *Proceedings of the International Computer Music Conference (ICMC)*, Beijing, China, Oct., 1999, pp. 299–304.
- [2] D. Jaffe, N. Porcaro, G. Scandalis, J. Smith, and T. Stilson, “SynthBuilder: a graphical real-time synthesis, processing and performance system,” in *Proceedings of the International Computer Music Conference (ICMC)*, Banff, Canada, 1995, pp. 61–64.

⁸JACK Audio Connection Kit: <http://jackaudio.org>.

⁹Open Sound Control (OSC) is a content format for messaging among computers, sound synthesizers and other multimedia devices.

FAUST file name	C++ code #declarations	FAUST code #declarations	Size gain for #lines	C++ code #lines	FAUST code no. lines	Size gain for #lines
blowBottle.dsp	74	30	59.5%	237	54	77.2%
blowHole.dsp	131	66	49.6%	373	104	72.1 %
bowed.dsp	92	45	51.1%	274	69	74.8%
brass.dsp	90	36	60%	272	63	76.8%
clarinet.dsp	78	35	55.1%	255	60	76.5%
flutestk.dsp	109	43	60.6%	309	70	77.3%
modalBar.dsp *	63	37	42.3%	217	78	64%
saxophony.dsp	98	42	57.1%	308	69	77.6%
sitar.dsp	57	25	56.1%	193	42	78.2%
bars *	164	35	78.7%	396	70	82.3%
voiceForm.dsp *	121	65	46.3%	325	109	66.5%
piano.dsp *	292	158	45.9%	750	246	67.2%

Table 1: Comparison of the code size of the STK object's C++ code with the FAUST code from FAUST-STK. The number of declarations was calculated in both cases by counting the number of semicolons in the code. In the case of the instruments where the file name is followed by the * sign, parameters data-bases were not taken into account.

- [3] Julius O. Smith III, *Physical Audio Signal Processing*, <https://ccrma.stanford.edu/~jos/pasp/>, Dec. 2010, online book.
- [4] Julius Smith and Romain Michon, "Nonlinear allpass ladder filters in FAUST," in *Proc. 14th Int. Conf. on Digital Audio Effects (DAFx-11)*, Paris, France, September 19–23, 2011.
- [5] J. Smith, "Efficient simulation of the reed-bore and bow-string mechanisms," in *Proceedings of the International Computer Music Conference (ICMC)*, The Hague, Holland, 1986, pp. 275–280.
- [6] J. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.
- [7] J. L. Kelly and C. C. Lochbaum, "Speech synthesis," *Proc. Fourth Int. Congress on Acoustics, Copenhagen*, pp. 1–4, September 1962, Paper G42.
- [8] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*, Springer Verlag, New York, USA, 1976.
- [9] K. Karplus and A. Strong, "Digital synthesis of plucked string and drum timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43–55, 1983.
- [10] M-P. Verge, *Aeroacoustics of Confined Jets with Applications to the Physical Modeling of Recorder-Like Instruments*, Ph.D. thesis, Eindhoven University, 1995.
- [11] X. Rodet, "One and two mass model oscillations for voice and instruments," in *Proceedings of the International Computer Music Conference (ICMC)*, Banff, Canada, 1995, pp. 207–214.
- [12] Julius O. Smith III, "Virtual electric guitars and effects using Faust and Octave," in *Proc. 6th Int. Linux Audio Conf. (LAC2008)*, <http://lac.linuxaudio.org/>, 2008, Paper and supporting website: http://ccrma.stanford.edu/realsimple/-faust_strings/, presentation overheads: <http://ccrma.stanford.edu/~jos/pdf/-LAC2008-jos.pdf>.
- [13] G. Essl and P. Cook, "Banded waveguides: Towards physical modeling of bowed bar percussion instruments," in *Proceedings of the International Computer Music Conference (ICMC)*, Beijing, China, Oct., 1999, pp. 321–324.
- [14] Julius O. Smith III and Perry R. Cook, "The second-order digital waveguide oscillator," in *Proc. 1992 Int. Computer Music Conf., San Jose*, 1992, pp. 150–153, Computer Music Association, <http://ccrma.stanford.edu/~jos/wgo/>.
- [15] J-M. Adrien, *Representations of Musical Signals*, chapter The missing link: Modal Synthesis, pp. 269–297, G. DePoli, A. Piccilli, and C. Roads Eds. MIT Press, Cambridge, MA, 1991.
- [16] J. L. Flanagan, "Voices of men and machines," *J. Acoust. Soc. of America*, vol. 51, pp. 1375–1387, Mar. 1972.
- [17] E. Castro Lopo (de), "libsndfile," Available at <http://www.mega-nerd.com/libsndfile/>, accessed March 01, 2011.
- [18] J. Smith and S. Van Duyne, "Commutated piano synthesis," in *Proceedings of the International Computer Music Conference (ICMC)*, Banff, Canada, 1995, pp. 319–326.
- [19] S. Sinclair, "Implementing the SynthBuilder piano in STK," Tech. Rep., McGill University, Canada, 2006.
- [20] A. Gräf, "faust2pd: Pd Patch Generator for Faust," Available at <http://docs.pure-lang.googlecode.com/hg/faust2pd.html>, accessed March 01, 2011.
- [21] E. Maestre-Gomez, *Modeling Instrumental Gesture: An Analysis/Synthesis Framework for Violin Bowing*, Ph.D. thesis, University Pompeu Fabra, 2009.
- [22] M. Gurevich and C. Chafe, "stk2pd," Available at <https://ccrma.stanford.edu/wiki/stk2pd>, accessed March 01, 2011.

ANALYSIS AND SIMULATION OF AN ANALOG GUITAR COMPRESSOR

Oliver Kröning, Kristjan Dempwolf and Udo Zölzer

Dept. of Signal Processing and Communications,
Helmut Schmidt University Hamburg
Hamburg, Germany
oliver.kroeningkristjan.dempwolfludo.zoelzer@hsuHH.de

ABSTRACT

The digital modeling of guitar effect units requires a high physical similarity between the model and the analog reference. The famous MXR DynaComp is used to sustain the guitar sound. In this work its complex circuit is analyzed and simulated by using state-space representations. The equations for the calculation of important parameters within the circuit are derived in detail and a mathematical description of the operational transconductance amplifier is given. In addition the digital model is compared to the original unit.

1. INTRODUCTION

In the field of guitar technology, certain products enjoy cultic status because of their unique auditory characteristics, like the MXR DynaComp. This guitar compressor pedal, created by MXR in the 1970's, was a very popular tool of achieving a fattened up sound with noticeable more sustain to lead guitar lines. The compression effect of the DynaComp is used to smooth out differences in volume between notes. Thereby it is a kind of volume controller that varies its internal gain to sustain the guitar sound.

In recent years a new trend has won recognition in music technology - the digital modeling and simulation of analog audio circuits. The advantage is the independence of cost-intensive, unreliable and often impractical analog technologies.

The state-space model has turned out to be a practicable tool to simulate non-linear audio systems with parametric components.

2. CIRCUIT ANALYSIS

The circuit of the MXR DynaComp is depicted in Fig. 1. For the sake of clarity we split this complex circuit into the four adequate blocks: (1) input circuit, (2) output circuit, (3) power supply and (4) the heart of the DynaComp, the operational transconductance amplifier (OTA). The simplified structure is given in Fig. 2 and shows the coupling between each stage.

The power supply block feeds the other blocks with two constant voltages $V_{batt} = 9\text{ V}$ and $V_{bias} = 2.93\text{ V}$. This subcircuit has no further effect on the audio signals and is neglected in the following considerations.

2.1. Input Stage

The input stage buffers the input signal and provides two signals to the differential inputs of the OTA, which are commensurate to the input signal.

Input capacitance C_3 isolates the internal biased DC-level from the 0 V DC-level of the guitar. Transistor Q_1 is used as a buffer

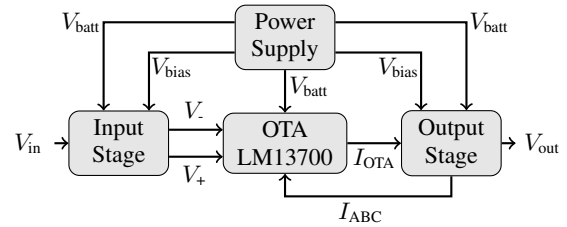


Figure 2: Simplified block diagram of the circuit.

and provides a low-impedance signal at its emitter. This buffered signal is set by capacitance C_2 to the DC-level of the OTA and is then routed to the inverting input. The signal is also routed to the non-inverting input with a potential drop at the potentiometer causing a signal-dependent difference between differential inputs.

2.2. OTA

The operational transconductance amplifier LM13700 is depicted in Fig. 3. It has a pair of differential inputs, a single output and one controllable gain input. The chip is completely composed of transistors and diodes. The task of the OTA is to produce an amplified output current depending on the differential input voltages. The gain of the OTA is variably controlled by the amplifier bias current I_{ABC} . Detailed information can be found in [1] and [2].

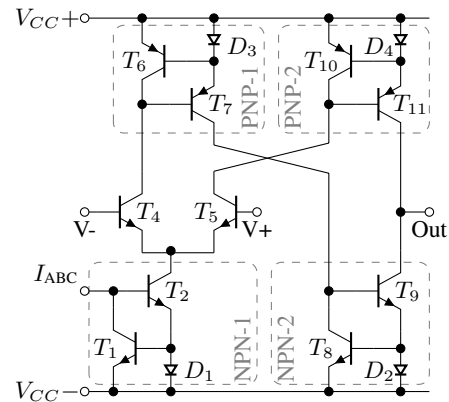


Figure 3: Simplified circuit of the OTA LM13700.

To find an analytical description of the OTA, it is necessary to simplify the integrated circuit. In addition to the differential amplifier, the integrated circuit can be expressed as an subtraction circuit

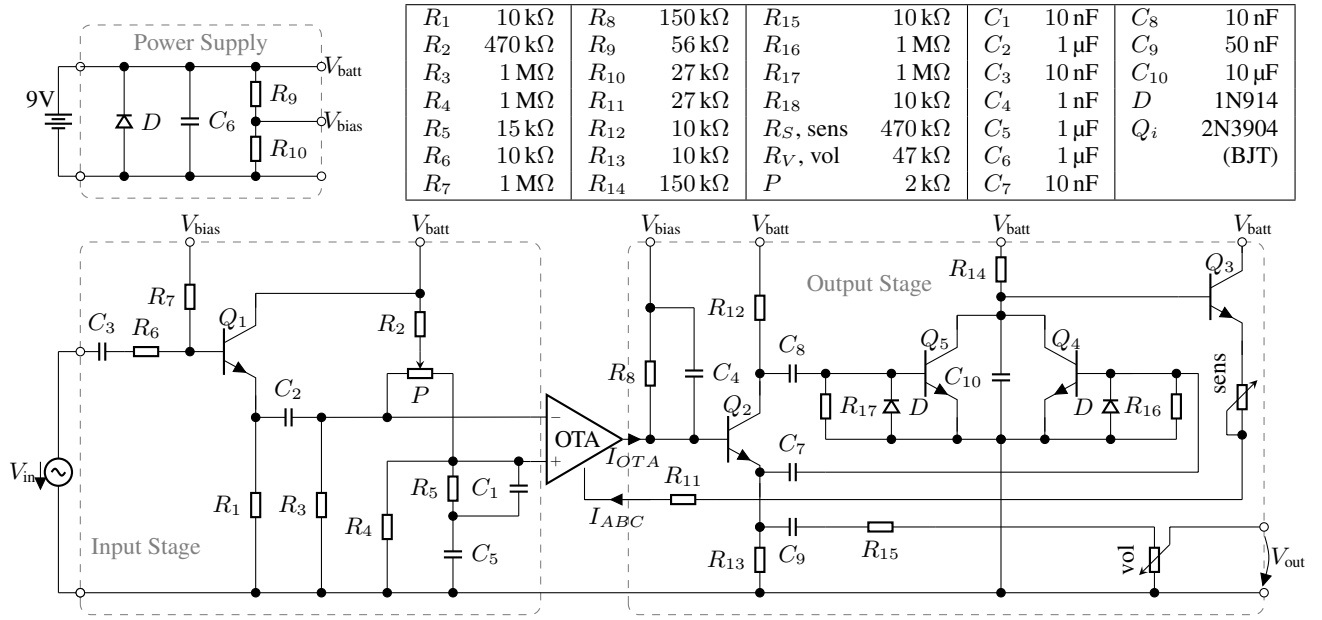


Figure 1: Schematic of the MXR DynaComp and component values.

composed of two NPN-current mirrors and two PNP-current mirrors. In the following we assume that the reference current and the output current are equal [3].

Since I_{ABC} is the reference current of the first NPN-current mirror, the same current has to be pulled out of the differential amplifier. Disregarding the base currents we obtain

$$I_{ABC} = I_{C,T4} + I_{C,T5} \quad (1)$$

with the collector currents of T_4 and T_5 composing the differential amplifier. Expressing these currents by the approximation

$$I_C = I_S \cdot e^{\frac{V_{BE}}{V_T}}, \quad (2)$$

with thermal voltage $V_T \approx 26$ mV, we consider the ratio

$$\frac{I_{C,T4}}{I_{C,T5}} = \frac{e^{\frac{V_{BE,T4}}{V_T}}}{e^{\frac{V_{BE,T5}}{V_T}}} = e^{\frac{V_{BE,T4} - V_{BE,T5}}{V_T}} \quad (3)$$

assuming that the saturation currents I_S of T_4 and T_5 are equal. With $V_D = V_{BE,T4} - V_{BE,T5} = V_- - V_+$ and equations (1) and (3) we obtain two expressions for the collector currents

$$I_{C,T4} = \frac{I_{ABC}}{e^{-\frac{V_D}{V_T}} + 1}, \quad I_{C,T5} = \frac{I_{ABC}}{e^{\frac{V_D}{V_T}} + 1}. \quad (4)$$

Transforming these terms using [4]

$$\frac{2}{1 + e^{-x}} = 1 + \frac{1 - e^{-x}}{1 + e^{-x}} = 1 + \tanh \frac{x}{2} \quad (5)$$

the collector currents result in

$$I_{C,T4} = \frac{I_{ABC}}{2} \left(1 + \tanh \frac{V_D}{2V_T} \right) \quad (6)$$

$$I_{C,T5} = \frac{I_{ABC}}{2} \left(1 - \tanh \frac{V_D}{2V_T} \right). \quad (7)$$

Tracking the collector currents through the current mirrors, the output current of the OTA is

$$\begin{aligned} I_{OTA} &= I_{C,T5} - I_{C,T4} = -I_{ABC} \cdot \tanh \frac{V_D}{2V_T} \\ &= -I_{ABC} \cdot \tanh \frac{V_- - V_+}{2V_T}. \end{aligned} \quad (8)$$

2.3. Output Stage

The output current of the OTA is the input variable of the output stage. This stage is a circuit controlled by the signal level of I_{OTA} applying the required gain to the OTA by feeding back the amplifier bias current I_{ABC} . Another task is to derive the output voltage V_{out} as the output signal of the DynaComp circuit.

The output of the OTA is attached to a high frequency roll-off composed of R_8 and C_4 and to transistor Q_2 . Q_2 performs two tasks - firstly it buffers the output signal and secondly it inverts the phase to follow the envelope of the signal. Both the emitter and the collector current, provide out-of-phase signals to a rectifier-filter arrangement. The negative parts of both signals are earthed by the diodes attached to the base of Q_5 and Q_4 . The base currents of these two transistors, derived from the emitter and collector currents of Q_2 , follow the envelope by controlling the voltage across capacitance C_{10} . This voltage represents an inversion of the input signal, i.e. higher signals of I_{OTA} causing higher base currents of Q_4 and Q_5 . Thus the voltage across C_{10} is pulled down because of the current flowing out of R_{14} divides into the collector currents of Q_4 and Q_5 . If there are small signals at the input, voltage V_{C10} rises - with no signal at the input it rises nearly to $V_{Batt} = 9$ V. Voltage V_{C10} controls I_{ABC} , which is adjustable by the 470 k Ω -potentiometer, as depicted in Fig. 4.

As a result, the amplifier bias current rises and amplifies the gain of the OTA in case of a decreasing input signal. The output

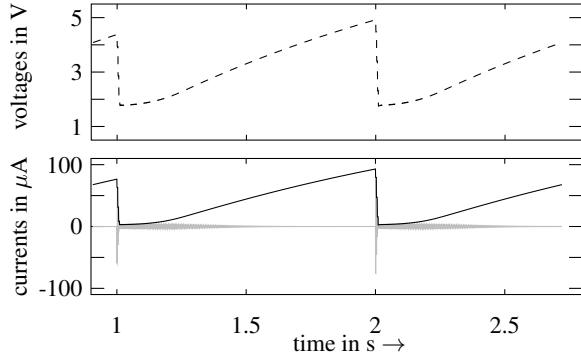


Figure 4: Stimulation with exponentially decreased sine burst. Capacitor state V_{C10} (dashed), currents I_{OTA} (gray) and I_{ABC} (black).

voltage of the circuit is tapped from the emitter of Q_2 . It is also adjustable by a 47 k Ω -potentiometer.

3. STATE-SPACE MODELS

The state-space representation is a common tool to describe physical systems, especially to simulate non-linear systems with changeable parameters. This method models the system as a set of input, \mathbf{u} , output, \mathbf{y} , and state variables, \mathbf{x} and $\dot{\mathbf{x}}$. The relation between them are based on network theory basics and formed into first-order differential equations. The equations used in this paper and the discretization are derived in detail in [5]. The differential equations describing continuous non-linear systems are

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{B} \cdot \mathbf{u}(t) + \mathbf{C} \cdot \mathbf{i}(\mathbf{v}(t)) \quad (9)$$

$$\mathbf{y}(t) = \mathbf{D} \cdot \mathbf{x}(t) + \mathbf{E} \cdot \mathbf{u}(t) + \mathbf{F} \cdot \mathbf{i}(\mathbf{v}(t)) \quad (10)$$

$$\mathbf{v}(t) = \mathbf{G} \cdot \mathbf{x}(t) + \mathbf{H} \cdot \mathbf{u}(t) + \mathbf{K} \cdot \mathbf{i}(\mathbf{v}(t)). \quad (11)$$

The number of independent energy storage elements defines the number of state variables in vector \mathbf{x} and $\dot{\mathbf{x}}$, typically the voltages across capacitors. The discrete-time system is obtained by using the trapezoidal rule with sampling interval T . The substitution $\mathbf{x}_c(n) = \frac{T}{2} \left(\left(\frac{2}{T} \mathbf{I} + \mathbf{A} \right) \mathbf{x}(n) + \mathbf{B} \mathbf{u}(n) \right)$ serves the discrete state-space model with

$$\mathbf{x}_c(n) = \bar{\mathbf{A}} \cdot \mathbf{x}_c(n-1) + \bar{\mathbf{B}} \cdot \mathbf{u}(n) + \bar{\mathbf{C}} \cdot \mathbf{i}(\mathbf{v}(n)) \quad (12)$$

$$\mathbf{y}(n) = \bar{\mathbf{D}} \cdot \mathbf{x}_c(n-1) + \bar{\mathbf{E}} \cdot \mathbf{u}(n) + \bar{\mathbf{F}} \cdot \mathbf{i}(\mathbf{v}(n)) \quad (13)$$

$$\mathbf{v}(n) = \bar{\mathbf{G}} \cdot \mathbf{x}_c(n-1) + \bar{\mathbf{H}} \cdot \mathbf{u}(n) + \bar{\mathbf{K}} \cdot \mathbf{i}(\mathbf{v}(n)). \quad (14)$$

To solve the non-linear equations given in (14) we use the damped Newton algorithm. Function $\mathbf{F}(\mathbf{v}(n))$ and its Jacobian $\mathbf{J}(\mathbf{v}(n))$ are required in this algorithm. To obtain a non-linear relation between the transistor currents $\mathbf{i} = (I_B, I_E)$ and the transistor voltages $\mathbf{v} = (v_{BE}, v_{BC})$ we use the Ebers-Moll equations

$$I_B = I_{ES} \frac{1}{1 + \beta_F} \left(e^{\frac{v_{BE}}{V_T}} - 1 \right) + I_{CS} \frac{1}{1 + \beta_R} \left(e^{\frac{v_{BC}}{V_T}} - 1 \right) \quad (15)$$

$$I_E = -I_{ES} \left(e^{\frac{v_{BE}}{V_T}} - 1 \right) + I_{CS} \frac{\beta_R}{1 + \beta_R} \left(e^{\frac{v_{BC}}{V_T}} - 1 \right). \quad (16)$$

4. SIMULATION

4.1. Input Stage

To simulate the input stage, we just have to use the algorithm of the discrete state-space model. Because there are no variations from the procedure explained in Section 3, we deal briefly with this part. With the conventions for currents through capacitances and transistors $i_C = C \cdot \dot{u}_C$ and $I_C + I_B + I_E = 0$ we obtain the system matrices by using Kirchhoff's circuit laws. Firstly all voltages across resistors have to be expressed by non-linear transistor elements, known values and/or capacitor states. Afterwards a mesh analysis has to be accomplished to find adequate meshes to express the transistor voltages, the output and the whole system. We find the mesh equations

$$0 = V_{in} - V_{R1} - V_{BE1} + V_{R6} + V_{C3}$$

$$0 = V_{R4} - V_{C5} - V_{C1}$$

$$0 = V_{R3} - V_{R4} + V_{Pb} + V_{Pa}$$

$$0 = V_{R2} + V_{Pa} + V_{C2} - V_{BE1} + V_{BC1}.$$

To gain the dependencies on the different parameters and to build the system matrices we have to solve these equations for \dot{v}_{C_i} . After this the discretization has to be operated and non-linear equations have to be solved. Since there are two output voltages, V_- and V_+ , the discrete output $\mathbf{y}(n)$ consists of two entries.

4.2. Output Stage

The simulation of the output stage is more complicated, because of the feedback of the amplifier bias current. This feedback causes linear dependencies in system matrix $\bar{\mathbf{K}}$, which has to be inverted to calculate the non-linear transistor currents. This problem is known as a delay-free loop which can be eliminated by using, for example, the K-method [6]. In our case we decided to simplify the feedback loop as a one sample delay.

A method to solve this problem is to modify the output stage by neglecting the base current of Q_3 . In addition Q_3 is replaced by a voltage-controlled voltage-source V_{C10} and a current-controlled current source with a non-linear internal resistance R_{ABC} . V_{C10} serves the same voltage as the voltage drop at capacitance C_{10} to control I_{ABC} as depicted in Fig. 4. Another problem is the unknown voltage drop at the ABC-input of the OTA, V_{ABC} , which is necessary to set up a mesh equation for the calculation of I_{ABC} . We obtain an expression for V_{ABC} by observing current mirror NPN-1 in Fig. 3 with

$$V_{ABC} = V_{BE,T2} + V_{D1}. \quad (17)$$

Knowing the electrical properties of the disposed elements within the integrated circuit, we can use the Shockley-equations with approximation (2) for the collector current of T_2 and diode current of D_1 to express I_{ABC} by

$$I_{ABC} \approx I_{C,T2} = I_{S,T2} \cdot e^{\frac{V_{BE,T2}}{V_T}} \approx I_{D1} = I_{S,D1} \cdot e^{\frac{V_{D1}}{V_T}} \quad (18)$$

with the saturation currents $I_{S,T2}$ and $I_{S,D1}$. After transposing and inserting these equations in (17), we get

$$V_{ABC} = V_t \cdot \left(\ln \left(\frac{I_{ABC}}{I_{S,T2}} \right) + \ln \left(\frac{I_{ABC}}{I_{S,D1}} \right) \right). \quad (19)$$

The voltage drop across R_{ABC} , \bar{V} , is defined by V_{ABC} and the base-emitter voltage of the replaced transistor Q_3 , V_{BE3} ,

$$\bar{V} = V_t \cdot \left(\ln \left(\frac{I_{ABC}}{I_{S,T2}} \right) + \ln \left(\frac{I_{ABC}}{I_{S,D1}} \right) + \ln \left(\frac{I_{ABC}}{I_{S,Q3}} \right) \right) \quad (20)$$

With the mesh equation

$$\bar{V} = V_{C10} - V_{R_S} - V_{R11} = V_{C10} - I_{ABC} \cdot (R_S - R_{11}), \quad (21)$$

I_{ABC} can be numerically calculated by solving

$$I_{ABC} = \sqrt[3]{I_{S,T2} \cdot I_{S,D1} \cdot I_{S,Q3} \cdot e^{\frac{V_{C10} - I_{ABC} \cdot (R_S - R_{11})}{V_t}}}. \quad (22)$$

Thus the simulation of the output stage can be done straight forward by using the conventions made in section 4.1 and the general algorithm of the discrete state-space model. The mesh analysis for the system is set up by

$$\begin{aligned} 0 &= -V_{R13} + V_{C7} + V_{BE4} \\ 0 &= -V_{Batt} + V_{R12} - V_{C8} + V_{BE5} \\ 0 &= -V_{Batt} + V_{R14} + V_{C10} \\ 0 &= -V_{bias} + V_{C4} + V_{BC2} - V_{C8} + V_{BC5} + V_{C10} \\ 0 &= V_{R8} - V_{bias} + V_{BE2} - V_{C9} - V_{R15} + V_{R_{vol1}} + V_{R_{vol2}} \end{aligned}$$

and provides on the one hand the output voltage, $V_{out} = V_{R_{vol2}}$, and on the other hand the voltage drop across C_{10} to calculate via (22) and (8) the new values of I_{ABC} and I_{OTA} [7]. The state-space vectors are

$$\dot{\mathbf{x}}(t) = [\dot{v}_{C4}(t) \dot{v}_{C7}(t) \dot{v}_{C8}(t) \dot{v}_{C9}(t) \dot{v}_{C10}(t)]^T, \quad (23)$$

$$\mathbf{u}(t) = [I_{OTA}(t) V_{bias}(t) V_{Batt}(t)]^T, \quad (24)$$

$$\mathbf{i}(t) = [I_{B2} I_{E2} I_{B4} I_{E4} I_{B5} I_{E5}]^T. \quad (25)$$

To simplify the circuit and to reduce computational costs, it is practical to enhance the Ebers-Moll-Model by connecting the diodes attached to Q_4 and Q_5 in parallel to the base-emitter junction. Thus equation 15 has to be modified [7].

5. RESULTS AND DISCUSSION

To evaluate the performance of the state-space model, the output data has been compared to the original MXR DynaComp as reference by stimulating both with the same test signals.

Fig. 5 displays the dynamic characteristics of both models using the same settings - sensitivity: maximum, output: maximum.

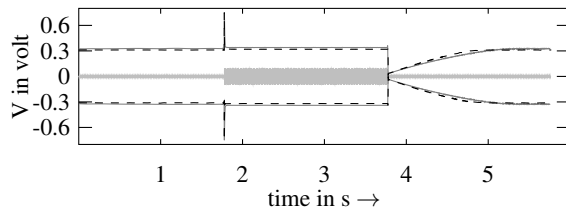


Figure 5: *Dynamic characteristic. Input signal with 10/100 mV (gray), envelopes of measurement (dashed) and simulation.*

For audio compressors it is useful to analyze the static behavior. Thus the static characteristics of the digital model is compared

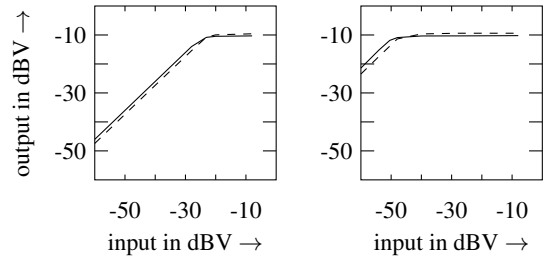


Figure 6: *Static curves of simulation (dashed) and measurement (solid). On the left: sensitivity = 0, right: sensitivity = 100.*

to the original pedal by showing the input-output-relation using different settings for sensitivity in Fig. 6. Both, the static and the dynamic characteristics, illustrate a good similarity. There are differences in the attack and release behavior which can be explained by component tolerances in the measured system and by approximations made during the derivation of the discrete system.

In addition some sound clips of electric guitar playing are available on our homepage

<http://ant.hsu-hh.de/dafx2011/compressor>

6. CONCLUSION

This paper presented a state-space model of the MXR DynaComp for a digital simulation of its sustaining and dynamic range controlling effect. The analog circuit was analyzed and equations for the calculation of the OTA output current and the amplifier bias current were derived in detail. The algorithm of the derivation of the discrete state-space matrices was introduced.

The comparison between the model and the reference showed a good match, too, although a lot of approximations had been made to develop a functional state-space description of the DynaComp.

7. REFERENCES

- [1] R. Marston, "Understanding and using ota op-amp ics, part 1," *Nuts and Volts*, pp. 58–62, April 2003.
- [2] R. Marston, "Understanding and using ota op-amp ics, part 2," *Nuts and Volts*, pp. 70–74, May 2003.
- [3] E.M. Zumchak, "A short discussion of the operational transconductance amplifier (ota)," [Online] Retrieved on 17th March 2011, February 1999.
- [4] U. Tietze and C. Schenk, *Halbleiterschaltungstechnik*, Springer Verlag, Berlin, Heidelberg, 12th edition, 2002.
- [5] K. Dempwolf, M. Holters, and U. Zölzer, "Discretization of parametric analog circuits for real-time simulations," in *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10 2010.
- [6] G. Borin, G. De Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," in *Proc. of the IEEE Transactions on Speech and Audio Processing*, Sept. 2000.
- [7] O. Kröning, "Analysis and simulation of an analog guitar compressor," B.s. thesis, Helmut Schmidt University Hamburg, 2011.

A SINGLE-AZIMUTH PINNA-RELATED TRANSFER FUNCTION DATABASE

Simone Spagnol,

Department of Information Engineering
Università degli Studi di Padova
Padova, Italy
spagnols@dei.unipd.it

Marko Hiipakka,

Aalto University School of Electronical Engineering
Department of Signal Processing and Acoustics
Espoo, Finland
Marko.Hiipakka@aalto.fi

Ville Pulkki,

Aalto University School of Electronical Engineering
Department of Signal Processing and Acoustics
Espoo, Finland
Ville.Pulkki@aalto.fi

ABSTRACT

Pinna-Related Transfer Functions (PRTFs) reflect the modifications undergone by an acoustic signal as it interacts with the listener's outer ear. These can be seen as the pinna contribution to the Head-Related Transfer Function (HRTF). This paper describes a database of PRTFs collected from measurements performed at the Department of Signal Processing and Acoustics, Aalto University. Median-plane PRTFs at 61 different elevation angles from 25 subjects are included. Such data collection falls into a broader project in which evidence of the correspondence between PRTF features and anthropometry is being investigated.

1. INTRODUCTION

Anthropometric features of the human body have a key role in Head-Related Transfer Function (HRTF) characterization. While non-individualized HRTFs represent a cheap and straightforward mean of providing 3-D perception in headphone reproduction, listening to non-individualized spatialized sounds may likely result in evident sound localization errors such as incorrect perception of elevation, front-back reversals, and lack of externalization [1], especially when head tracking is not utilized in the reproduction [2]. On the other hand, individual HRTF measurements on a significant number of subjects may be both expensive and inconvenient.

Structural modeling of HRTFs ultimately represents an attractive solution to these shortcomings. As a matter of fact, if one isolates the contributions of the listener's head, pinnae, ear canals, shoulders, and torso to the HRTF in different subcomponents - each accounting for some well-defined physical phenomenon - then, thanks to linearity, one can reconstruct the global HRTF from a proper combination of all the considered effects. Relating each subcomponent's temporal and/or spectral features (in the form of digital filter parameters) to the corresponding anthropometric quantities would then yield a HRTF model which is both economical and individualizeable [3].

Following such structural assumption, the present work focuses on the pinna contribution to the HRTF. It is undoubted that the pinna plays a primary part in the perception of source elevation; even so, the relation between acoustic phenomena due to

the pinna - mainly resonances and sound reflections [4] - and anthropometry is yet not fully understood. In [5] a promising correspondence between reflection points on pinna surfaces and frequencies of notches occurring in the high-frequency range of the HRTF spectrum was informally found by analyzing median-plane responses from the CIPIC database [6] along with photos of four subjects' pinnae. Still, in order for a more extensive and accurate analysis to be carried out, an alternative database is needed, the most relevant reasons being:

- the presence in the CIPIC HRTFs of the head and shoulders' contributions, which cannot be fully eliminated a posteriori;
- the need of highly detailed photographs of the subjects' pinnae for an effective image processing method that extracts the possible reflection contours to be designed;
- the public unavailability of the remaining subjects' pinnae photographs, necessary to perform such an analysis.

This paper presents such a database, which we refer to as Pinna-Related Transfer Function (PRTF) Database, primarily focusing on the choices and tools through which the final responses were collected. Furthermore, some early results on the so obtained PRTFs' features will be sketched. All of the following work was carried out at the Department of Signal Processing and Acoustics, Aalto University. The database, accompanied by detailed photographs of the subjects' pinnae and the measurement setup, is publicly downloadable from the first author's website at http://www.dei.unipd.it/~spagnols/PRTF_db.zip as a .zip archive.

2. MEASUREMENT PROCEDURE AND APPARATUS

In an ideal situation, the PRTF is the response of the pinna mounted on an infinite plane [7]. In our measurements, an ad hoc pinna isolation device that approximates the ideal case was built and used. The test subjects' torsos and shoulders were isolated by a $1\text{-m} \times 1\text{-m}$, 15-mm thick wooden board having a 24-cm-diameter circular hole in the middle of it that approximately fits the size of the human head. A polycarbonate sheet with grinded edges and a 6-cm-diameter circular hole in the middle was fixed with a dozen



Figure 1: The pinna isolation device used for PRTF measurements.



Figure 2: Subject position during the measurements. On top left, the boom-controlled loudspeaker used for sweep reproduction.

flat head screws to the board in order to completely cover the hole for the head while letting the subjects' right pinnae come out of the other side of it (see Figure 1). Furthermore, a thick layer of foam with a head-profile-shaped cut in the middle was glued to the upper side of the board with the purpose of adding comfort to the subjects. A piece of such layer could be taken off accordingly with the specific subject's build.

The isolation device was brought right in the middle of an anechoic chamber and placed over an acoustically transparent, one meter high cylindrical metallic fence having 1.75mm thread width in order to avoid reflections from prospective table legs. A controlled boom mounted on the room's ceiling had the purpose of moving the sound source (a Genelec 8030A loudspeaker) along a circumference centered in the pinna hole and laying on the plane parallel to the isolation device. The loudspeaker was positioned upside down, so that the woofer was at the level of the forementioned plane while the tweeter was under it, allowing high-frequency components to directly join the pinna hole without reflecting on the border of the isolation device. Furthermore, the distance between the loudspeaker and the pinna hole was approximately 1.6 m, so we can assume the incident wave to be plane for frequencies above 3 kHz (the loudspeaker's crossover frequency). This assumption may not be guaranteed below 3 kHz, yet the relative little importance of pinna features below this threshold makes this problem negligible. Since the boom was not acoustically transparent and other loudspeakers were fixed to the chamber's walls, let us label the environment as low-echoic rather than anechoic. In spite of this, as Subsection 2.1 will mention, all the data will be adequately windowed so as to discard reflections occurring on the



Figure 3: Subject 08's right pinna.

room's equipment.

25 subjects (18 men and 7 women), mostly students and staff of Aalto University, participated to the measurements. A Knowles FG-23329 microphone carefully stuffed in the middle of a hollow earplug was placed inside the right ear canal of each subject in turn. Then, the subject was asked to stand in front of one side of the panel (eventually with the help of a pedestal to let his waist reach the level of the isolation device), bend 90 degrees forwards and lay his head on the right side in order to let his pinna pass the hole (see Figure 2). The required 90° head-neck rotation could be reached thanks to the thick layer of foam which allowed the right shoulder to sink at a lower level than the left. This way, the plane spanned by the loudspeaker's rotation approximately corresponded to the subject's median plane. The pinna position was then adjusted both by instructing the subject on how to move his head and by manual intervention through a big hole in the fence. Finally, vertical orientation was adjusted by manually rotating the subject's head to let his ear axis point at a precise mark on one of the chamber's walls. Subjects were told to remain as still as possible, yet their movements were not monitored during the actual measurement session.

The responses were measured via the logarithmic sweep (or logsweep) method [8]. The used sine sweep had 48 kHz sampling frequency, 1 s duration, and spanned the frequency range from 20 Hz to 22 kHz. By controlling the boom rotation and sweep reproduction from a Max/MSP patch running on a workstation just outside the anechoic chamber, sweep responses for 61 different elevation angles were recorded at 48 kHz sampling frequency in approximately six minutes' time. The selected elevation angles, considering the interaural-polar coordinate system (see [6]), spanned the range from -60 to 240 degrees at 5-degree steps. The boom constantly rotated during the measurements, hence high frequencies were measured from a slightly different elevation than low frequencies. However, since the angular speed was almost constantly less than one degree per second, the impact on measurements looks negligible.

In addition, free-field responses were taken by placing the microphone-stuffed earplug inside of a small foam cut, positioning it in the middle of the pinna hole of the isolation device, and repeating the measurement procedure in the same way as for the test subjects.

Pictures of the subjects' right pinnae were also taken before or after the measurements (see e.g. Figure 3). The distance and orientation of the camera with respect to the pinna was kept as

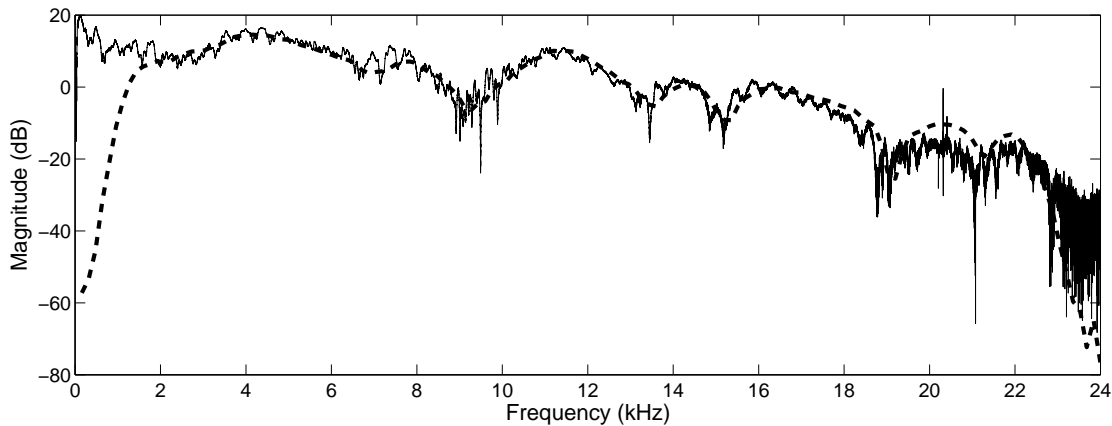


Figure 4: Original sweep magnitude response (solid line) and post-processed PRTF magnitude (thick dashed line).

constant among subjects as possible through the help of a tripod. Also, each subject's pinna height (variable d_5 in [6]) was measured and tracked down for resizing purposes. This information, along with each subject's sex and evidenced anomalies in the experiment with respect to the optimal situation, can be found in the online database. As for anomalies, Subject 06's pinna did not completely pass the hole, Subject 13 had a piercing on the helix which could not be taken off, and Subject 18 had the earplug slightly displaced at the end of the measurements.

2.1. Data post-processing

According to the logsweep method, inverse filtering was performed on the measured sweeps (including free-field sweeps) in order to obtain the corresponding impulse responses. Specifically, the inverse response of the excitation signal was first computed and then low-passed and high-passed with fifth-order digital Butterworth filters to compensate for the original zero sound pressure level below 20 Hz and above 22 kHz in the sweep signal. Since the pinna has no effect below 3 kHz and sounds above 15-20 kHz are hardly perceptible by humans we let the high-pass and low-pass Butterworth filters' cutoff frequency be loose, that is 1.2 and 21.6 kHz respectively. Hence, each impulse response was calculated by convolving such band-passed inverse filter with the measured sweep.

Subsequently, a 300-sample Hann window was applied to each impulse response with the aim of cutting off late reflections possibly occurring on the subject's legs, the pedestal, or the room equipment. The window was centered in the first positive peak p exceeding a heuristic amplitude threshold in the impulse response, so that the windowed impulse lasts approximately 3 ms from p .

Finally, free-field compensation of the subjects' impulse responses had to be performed. To this end, for each elevation e , a 10th-order minimum-phase IIR filter which approximates the magnitude of the inverse free-field response at source elevation e was designed through the least-squares fit procedure provided by the Yule-Walker method of ARMA spectral estimation [9]. As we expected, all free-field responses had similar and almost flat - except for a ripple around 2.5 kHz probably due to the loudspeaker's crossover frequency - magnitude plots, with no tangible diffraction occurring on the wooden board. This result certifies the transparency of the measurement setup. Straightforward filtering of the

subject's impulse response at elevation e through the so built IIR filter gave the free-field compensated, final pinna-related impulse response (PRIR) that is stored in the database.

Figure 4 shows the magnitude plots of an original recorded sweep and the corresponding post-processed PRTF. It can be clearly seen how the general notch/resonance structure of the typical PRTF is preserved, excluding the very upper and lower frequency ranges which are, however, not of interest to us.

3. EARLY RESULTS AND DISCUSSION

It should be mentioned that, since data was collected for a single azimuth value only, there is no guarantee that integrating a future pinna model based on these responses in a complete structural model would give an appropriate representation of the HRTF. In other words, the PRTF for elevation e and azimuth 0° may have a totally different look than the PRTF for elevation e and e.g. azimuth 60° . However, informal inspection of different HRTF sets revealed how generally there is no pronounced variation in median-plane reflection and resonance patterns across the angular range when the azimuth's absolute value is increased from 0° up to about 30° . Hence we may assert that under the assumption that the source moves in the vicinity of the median plane, pinna effects solely depend on source elevation.

Through direct inspection of the PRTF magnitude plots of all 25 subjects, a couple of observations can be made. First, when the source is ahead of the frontal plane (in our case when $-60^\circ \leq e < 90^\circ$), the PRTF behaviour is quite complex and greatly varies from subject to subject. However, commonly known features evidenced in previous works on PRTFs [10, 11], such as the 4-kHz omnidirectional resonance mode and the notch whose frequency (6 – 10 kHz) increases with elevation, appear in the vast majority of subjects (see e.g. Subject 08 in Figure 5). In some cases (e.g. Subject 15), however, the reflection structure is unclear, the magnitude plot presenting valleys which happen to be excessively shallow.

Secondly, while all PRTFs greatly differ among subjects when the source is ahead of the frontal plane, their behaviour is similar for all other elevations. Specifically, allowing some degree of approximation:

- for $90^\circ \leq e \leq 125^\circ$ the majority of PRTFs show a descending magnitude plot with one major resonance around

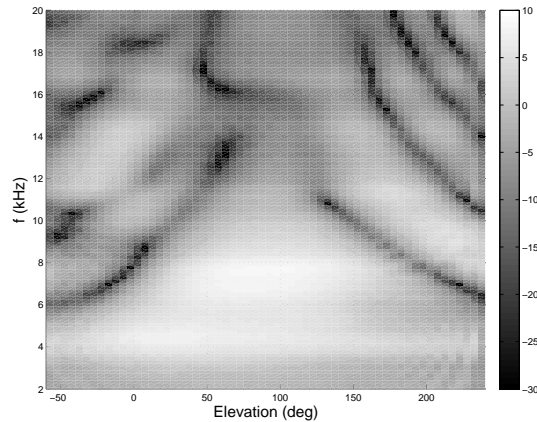


Figure 5: Magnitude plot (in dB) of Subject 08's PRTF.

7 kHz and no evident notches;

- from about $e = 130^\circ$ one frequency notch appears at around 10 kHz, eventually followed by others at higher frequencies when the source is about to cross the horizontal plane at $e = 180^\circ$ (this notch was found in [11] too);
- PRTFs for the last elevation angles, especially $e = 240^\circ$, show a more complex magnitude structure with 3 or more notches below 15 kHz (also reported in [11]).

These features can all be detected in Figure 5. The absence of evident notches when the source is above the listener may easily be attributed to the presence of the helix which “masks” the concha, evading direct reflections on it. Conversely, the presence of complicated patterns at $e = 240^\circ$ comparable to those for sources ahead of the frontal plane may be both attributed to reflections on different pinna contours such as the upper part of the helix, the tragus or the crus helias, or to possible unwind reflections on the subject's legs.

Finally, even after post-processing some PRTFs still present a “noisy” spectrum. This artifact may likely be associated to subjects' slight movements during the sweep reproduction or to a rattling noise coming from the metallic fence which was reported by a few subjects right after their measurement session. However, besides being isolated cases only, the main features of PRTFs remain preserved.

4. CONCLUSIONS AND FUTURE WORK

A database of Pinna-Related Transfer Functions was presented in this paper. The measurement setup and procedure was described in details, along with the polishing operations applied to obtain the final PRTFs from the measured responses. The early results and assumptions traced in the last section need of course to be further investigated, especially for what concerns PRTF behaviour in the elevation range $-60^\circ \leq e \leq 90^\circ$ where pinna modifications happen in greater number. Future work includes adaptation of a separation algorithm [12] that extracts the reflective and resonant components from each PRTF to the present database in order to analyze each component separately and study the relation between its features and anthropometry, the final aim of such work being customization of a structural HRTF model.

5. ACKNOWLEDGMENTS

The first author would like to acknowledge the entire spatial sound research group at Aalto University for the constant support throughout setting up and performing this experiment, especially Javier Gomez Bolaños for his precious assistance in photography issues. This work has been supported by The Academy of Finland and by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC Grant agreement No. 240453.

6. REFERENCES

- [1] H. Møller, M. F. Sørensen, C. B. Jensen, and D. Hammer-shøj, “Binaural technique: Do we need individual recordings?,” *J. Audio Eng. Soc.*, vol. 44, no. 6, pp. 451–469, 1996.
- [2] W. R. Thurlow and P. S. Runge, “Effect of induced head movements on localization of direction of sounds,” *J. Acoust. Soc. Am.*, vol. 42, no. 2, pp. 480–488, August 1967.
- [3] C. P. Brown and R. O. Duda, “A structural model for binaural sound synthesis,” *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 5, pp. 476–488, 1998.
- [4] D. W. Batteau, “The role of the pinna in human localization,” *Proc. R. Soc. London. Series B, Biological Sciences*, vol. 168, no. 1011, pp. 158–180, August 1967.
- [5] S. Spagnol, M. Geronazzo, and F. Avanzini, “Fitting pinna-related transfer functions to anthropometry for binaural sound rendering,” in *IEEE International Workshop on Multimedia Signal Processing*, Saint-Malo, France, October 2010, pp. 194–199.
- [6] R. V. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, “The CIPIC HRTF database,” in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, USA, 2001, pp. 1–4.
- [7] R. V. Algazi, R. O. Duda, R. P. Morrison, and D. M. Thompson, “Structural composition and decomposition of HRTFs,” in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, USA, 2001, pp. 103–106.
- [8] S. Müller and P. Massarani, “Transfer-function measurement with sweeps,” *J. Audio Eng. Soc.*, vol. 49, no. 6, pp. 443–471, June 2001.
- [9] B. Friedlander and B. Porat, “The modified Yule-Walker method of ARMA spectral estimation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, no. 2, pp. 158–173, March 1984.
- [10] E. A. G. Shaw and R. Teranishi, “Sound pressure generated in an external-ear replica and real human ears by a nearby point source,” *J. Acoust. Soc. Am.*, vol. 44, no. 1, pp. 240–249, 1968.
- [11] Y. Kahana and P. A. Nelson, “Boundary element simulations of the transfer function of human heads and baffled pinnae using accurate geometric models,” *Journal of Sound and Vibration*, vol. 300, no. 3-5, pp. 552–579, 2007.
- [12] M. Geronazzo, S. Spagnol, and F. Avanzini, “Estimation and modeling of pinna-related transfer functions,” in *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, September 2010.

FAUST ARCHITECTURES DESIGN AND OSC SUPPORT.

D. Fober, Y. Orlarey, S. Letz

Grame
Centre national de création musicale
Lyon, France

{fober, orlarey, letz}@grame.fr

ABSTRACT

FAUST [Functional Audio Stream] is a functional programming language specifically designed for real-time signal processing and synthesis. It consists in a compiler that translates a FAUST program into an equivalent C++ program, taking care of generating the most efficient code. The FAUST environment also includes various *architecture* files, providing the glue between the FAUST C++ output and the host audio and GUI environments. The combination of architecture files and FAUST output gives ready to run applications or plugins for various systems, which makes a single FAUST specification available on different platforms and environments without additional cost. This article presents the overall design of the architecture files and gives more details on the recent OSC architecture.

1. INTRODUCTION

From a technical point of view FAUST¹ (*Functional Audio Stream*) is a functional, synchronous, domain specific language designed for real-time signal processing and synthesis. A unique feature of Faust, compared to other existing languages like Max, PD, SuperCollider, etc., is that programs are not interpreted, but fully compiled.

One can think of FAUST as a *specification language*. It aims at providing the user with an adequate notation to describe *signal processors* from a mathematical point of view. This specification is free, as much as possible, from implementation details. It is the role of the FAUST compiler to provide automatically the best possible implementation. The compiler translates FAUST programs into equivalent C++ programs taking care of generating the most efficient code. The compiler offers various options to control the generated code, including options to do fully automatic parallelization and take advantage of multicore machines.

The generated code can generally compete with, and sometimes even outperform, C++ code written by seasoned programmers. It works at the sample level, it is therefore suited to implement low-level DSP functions like recursive filters up to full-scale audio applications. It can be easily embedded as it is self-contained and doesn't depend of any DSP library or runtime system. Moreover it has a very deterministic behavior and a constant memory footprint.

From a syntactic point of view FAUST is a textual language, but nevertheless block-diagram oriented. It actually combines two approaches: *functional programming* and *algebraic block-diagrams*. The key idea is to view block-diagram construction as function

composition. For that purpose, FAUST relies on a *block-diagram algebra* of five composition operations (`:`, `~`, `<:`, `>:`) [1, 2].

We don't have the space to describe the language in details but as an example here is how to write a pseudo random number generator r in Faust²:

```
r = +(12345)~*(1103515245);
```

This example uses the recursive composition operator `~` to create a feedback loop as illustrated figure 1.

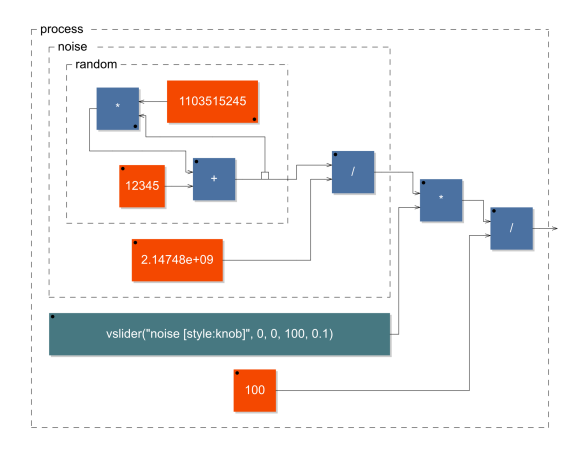


Figure 1: Block-diagram of a noise generator. This image is produced by the FAUST compiler using the `-svg` option.

Being a specification language the FAUST code says nothing about the audio drivers or the GUI toolkit to be used. It is the role of the architecture file to describe how to relate the dsp code to the external world. This approach allows a single FAUST program to be easily deployed to a large variety of audio standards (Max-MSP externals, PD externals, VST plugins, CoreAudio applications, Jack applications, etc.). In the following sections we will detail this architecture mechanism and in particular the recently developed OSC architecture that allows FAUST programs to be controlled by OSC messages.

2. FAUST SIGNAL PROCESSORS

A FAUST program denotes a signal processor implemented as an instance of a dsp class, defined as follows:

²Please note that this expression produces a signal $r(t) = 12345 + 1103515245 * r(t - 1)$ that exploits the particularity of 32-bits integer operations

¹<http://faust.grame.fr>

```

class dsp {
public:
    dsp() {}
    virtual ~dsp() {}
    virtual int getNumInputs() = 0;
    virtual int getNumOutputs() = 0;
    virtual void buildUserInterface(UI* ui) = 0;
    virtual void init(int samplingRate) = 0;
    virtual void compute(int len, float** in,
                        float** out) = 0;
};

```

The dsp object is central to the FAUST architectures design:

- `buildUserInterface` creates the user interface,
- `compute` is called by the audio architecture for the signal processing,
- `getNumInputs`, `getNumOutputs` provides information about the signal processor,
- `init` is called to initialize the sampling rate, which is typically done by the audio architecture.

3. AUDIO ARCHITECTURE FILES

A FAUST audio architecture is a glue between the host audio system and a FAUST module. It is responsible to allocate and release the audio channels and to call the FAUST `dsp::compute` method to handle incoming audio buffers and/or to produce audio output. It is also responsible to present the audio as non-interleaved float data, normalized between -1. and 1.

A FAUST audio architecture derives an *audio* class defined as below:

```

class audio {
public:
    audio() {}
    virtual ~audio() {}
    virtual bool init(const char* name, dsp*) = 0;
    virtual bool start() = 0;
    virtual void stop() = 0;
};

```

The API is simple enough to give a great flexibility to audio architectures implementations. The `init` method should initialize the audio. At `init` exit, the system should be in a safe state to recall the `dsp` object state.

Table 4 gives the audio architectures currently available for various operating systems.

4. GUI ARCHITECTURE FILES

A FAUST UI architecture is a glue between a host control layer and a FAUST module. It is responsible to associate a FAUST module parameter to a user interface element and to update the parameter value according to the user actions. This association is triggered by the `dsp::buildUserInterface` call, where the `dsp` asks a UI object to build the module controllers.

Since the interface is basically graphic oriented, the main concepts are *widget* based: a UI architecture is semantically oriented to handle active widgets, passive widgets and widgets layout.

A FAUST UI architecture derives an *UI* class (defined in appendix 10.1).

Audio system	Operating system
Alsa	Linux
Core audio	Mac OS X, iOS
Jack	Linux, Mac OS X, Windows
Portaudio	Linux, Mac OS X, Windows
OSC (see section 5.2)	Linux, Mac OS X, Windows
VST	Mac OS X, Windows
Max/MSP	Mac OS X, Windows
CSound	Linux, Mac OS X, Windows
SuperCollider	Linux, Mac OS X, Windows
PureData	Linux, Mac OS X, Windows
Pure[3]	Linux, Mac OS X, Windows

Table 1: FAUST audio architectures

4.1. Active widgets

Active widgets are graphical elements that control a parameter value. They are initialized with the widget name and a pointer to the linked value. The widget currently considered are `Button`, `ToggleButton`, `CheckButton`, `VerticalSlider`, `HorizontalSlider` and `NumEntry`.

A GUI architecture must implement a method

`addxxx (const char* name, float** zone, ...)` for each active widget. Additional parameters are available to `Slider` and `NumEntry`: the `init` value, the `min` and `max` values and the `step`.

4.2. Passive widgets

Passive widgets are graphical elements that reflect values. Similarly to active widgets, they are initialized with the widget name and a pointer to the linked value. The widget currently considered are `NumDisplay`, `TextDisplay`, `HorizontalBarGraph` and `VerticalBarGraph`.

A UI architecture must implement a method

`addxxx (const char* name, float** zone, ...)` for each passive widget. Additional parameters are available, depending on the passive widget type.

4.3. Widgets layout

Generally, a GUI is hierarchically organized into boxes and/or tab boxes. A UI architecture must support the following methods to setup this hierarchy :

```

openTabBox (const char* label)
openHorizontalBox (const char* label)
openVerticalBox (const char* label)
closeBox (const char* label)

```

Note that all the widgets are added to the current box.

4.4. Metadata

The FAUST language allows widget labels to contain metadata enclosed in square brackets. These metadata are handled at GUI level by a `declare` method taking as argument, a pointer to the widget associated value, the metadata key and value:

```
declare(float*, const char*, const char*)
```

Table 2 gives the UI architectures currently available.

UI	Comment
console	a command line UI
GTK	a GTK based GUI
Qt	a multi-platform Qt based GUI
FUI	a file based UI to store and recall modules states
OSC	see section 5.1

Table 2: FAUST UI architectures

5. OSC ARCHITECTURES

The OSC support opens the FAUST applications control to any OSC capable application or programming language. But it also transforms a full range of devices embedding sensors (wiimote, smart phones...) into physical interfaces for FAUST applications control, allowing a direct use as music instrument (which is in phase with the new FAUST physical models library adapted [4] from STK [5]).

The FAUST OSC architecture provides an UI architecture but also an audio architecture. This audio architecture runs at the OSC data stream rate, meaning that it allows to slow the audio computation down, up to frame by frame computation, and thus proposing a new and original way to make digital signal computation.

5.1. OSC GUI architecture

The OSC UI architecture transforms all the UI active widgets additions into an `addnode` call, ignores the passive widgets and transforms containers calls (`openxxxBox`, `closeBox`) into `opengroup` and `closegroup` calls.

5.1.1. OSC address space and messages

The OSC address space adheres strictly to the hierarchy defined by the `addnode` and `opengroup`, `closegroup` calls. It supports the OSC pattern matching mechanism.

A node expects to receive OSC messages with a single float value as parameter. This policy is strict for the parameters count, but relaxed for the parameter type: OSC int values are accepted and cast to float.

Two additional messages are defined to provide FAUST applications discovery and address space discoveries:

- the `hello` message: accepted by any module root address. The module responds with its root address, followed by its IP address, followed by the UDP ports numbers (listening port, output port, error port). See the network management section below for ports numbering scheme.
- the `get` message: accepted by any valid OSC address. The `get` message is propagated to every terminal node that responds with its OSC address and current values (value, min and max).

Example:

Consider the `noise` module provided with the FAUST examples:

- it sends `/noise 192.168.0.1 5510 5511 5512` in answer to a `hello` message,
- it sends `/noise/Volume 0.8 0. 1.` in answer to a `get` message.

5.1.2. Network management

The OSC architecture makes use of 3 different UDP port numbers:

- 5510 is the listening port number: control messages should be addressed to this port.
- 5511 is the output port number: answers to query messages are sent to this port.
- 5512 is the error port number: used for asynchronous errors notifications.

When the UDP listening port number is busy (for instance in case of multiple FAUST modules running), the system automatically looks for the next available port number. Unless otherwise specified by the command line, the UDP output port numbers are unchanged.

A module sends its name (actually its root address) and allocated ports numbers on the OSC output port on startup.

Ports numbers can be changed on the command line with the following options:

```
[-port | -output | -errport] number
```

The default UDP output streams destination is `localhost`. It can also be changed with the command line option

```
-dest address where address is a host name or an IP number.
```

5.2. OSC audio architecture

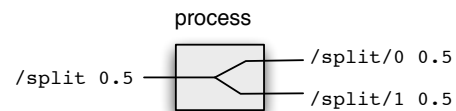
The OSC audio architecture provides audio input and output using OSC messages. It is not intended for real-time audio transportation due to the overhead introduced by the OSC coding. But, as we will explain, it provides a very useful and powerful mean to analyze and/or debug the behaviour of a Faust application.

Using this architecture, a FAUST module accepts arbitrary data streams on its root OSC address, and handles this stream as input interleaved signals. Each incoming OSC packet addressed to a module root triggers a computation cycle, where as much values as the number of incoming frames are computed.

The output of the signal computation is sent to the OSC output port as non-interleaved data to the OSC addresses `/root/n` where `root` is the module root address and `n` is the output number (indexed from 0). For example, consider a simple FAUST program named `split` and defined by:

```
process = _ <: _, _;
```

expected and generated OSC datagrams are illustrated in figure 2.

Figure 2: In and out OSC datagrams for the `split` module.

The OSC audio architecture provides a very convenient way to execute a signal processing at an arbitrary rate, allowing even to make step by step computation. Connecting the output OSC signals to Max/Msp or to a system like INScore³, featuring a powerful dynamic signals representation system, provides a close examination of the computation results.

³<http://inscore.sf.net>

6. OPEN ISSUES AND FUTURE WORKS

Generally, the labeling scheme for a GUI doesn't result in an optimal OSC address space definition. Moreover, there are potential conflicts between the FAUST UI labels and the OSC address space since some characters are reserved for OSC pattern matching and thus forbidden in the OSC naming scheme. The latter issue is handled with automatic characters substitutions. The first issue could be solved using the metadata scheme and will be considered in a future release.

Another issue, resulting from the design flexibility, relies on dynamic aggregation of multiple architectures covering the same domain: for example, it would be useful to embed both a standard and the OSC audio architecture in the same module and to switch dynamically between (for debugging purpose for example). That would require the UI to include the corresponding control and thus a mechanism to permit the UI extension by the UI itself would be necessary.

7. CONCLUSIONS

FAUST is a mature language for the design of signal processors. The FAUST architectures give the developer a handful of various binary outputs without additional cost. The architectures design, made for easy extension and combination, fits very well in the landscape of the evolving technologies. Adaptation to new hardware or software is simple and opens the door to all the existing FAUST programs. The recent OSC addition makes the connection between FAUST modules and the more and more ubiquitous hardware embedding sensors and usable as gestural controllers. FAUST architectures add a practical and ready to run dimension to the powerful FAUST language and additional contributions are welcome.

8. ACKNOWLEDGMENTS

The OSC architecture has been designed in the joint context of the ASTREE project [ANR-08-CORD-003] and the *Parallelism in Interactive Real-Time Signal Processing with FAUST and OSC* project made in collaboration with the CNMAT and supported by the France-Berkeley fund.

9. REFERENCES

- [1] Y. Orlarey, D. Fober, and S. Letz, "An algebra for block diagram languages," in *Proceedings of International Computer Music Conference*, ICMA, Ed., 2002, pp. 542–547.
- [2] Y. Orlarey, D. Fober, and S. Letz, *New Computational Paradigms for Computer Music*, chapter FAUST : an Efficient Functional Approach to DSP Programming, pp. 65–96, Editions DELATOUR FRANCE, 2009.
- [3] Albert Graef, "Signal processing in the pure programming language," in *Proceedings of the Linux Audio Conference LAC2009*, 2009.
- [4] R. Michon and J. O. Smith, "Faust-stk: a set of linear and non-linear physical models for the faust programming language.," in *submitted to DAFx 2011*, 2011.
- [5] P. Cook, "The synthesis toolkit (stk)," in *Proceedings of the International Computer Music Conference (ICMC)*, Beijing, China, Oct., 1999, pp. 299–304.

10. APPENDIX

10.1. UI class

```
class UI
{
public:
    UI() {}
    virtual ~UI() {}

    // -- active widgets
    virtual void addButton(const char* label, float* zone) = 0;
    virtual void addToggleButton(const char* label, float* zone) = 0;
    virtual void addCheckButton(const char* label, float* zone) = 0;
    virtual void addVerticalSlider(const char* label, float* zone,
                                  float init, float min, float max, float step) = 0;
    virtual void addHorizontalSlider(const char* label, float* zone,
                                     float init, float min, float max, float step) = 0;
    virtual void addNumEntry(const char* label, float* zone, float init,
                             float min, float max, float step) = 0;

    // -- passive widgets
    virtual void addNumDisplay(const char* label, float* zone, int precision) = 0;
    virtual void addTextDisplay(const char* label, float* zone,
                                const char* names[], float min, float max) = 0;
    virtual void addHorizontalBargraph(const char* label, float* zone,
                                       float min, float max) = 0;
    virtual void addVerticalBargraph(const char* label, float* zone,
                                     float min, float max) = 0;

    // -- widget's layouts
    virtual void openTabBox(const char* label) = 0;
    virtual void openHorizontalBox(const char* label) = 0;
    virtual void openVerticalBox(const char* label) = 0;
    virtual void closeBox() = 0;

    // -- metadata declarations
    virtual void declare(float*, const char*, const char*) {}
};
```

10.2. Available FAUST architectures

Audio system	Environment	OSC support
Linux		
Alsa	GTK, Qt	yes
Jack	GTK, Qt, Console	yes
PortAudio	GTK, Qt	yes
Mac OS X		
CoreAudio	Qt	yes
Jack	Qt, Console	yes
PortAudio	Qt	yes
Windows		
Jack	Qt, Console	yes
PortAudio	Qt	yes
iOS (iPhone)		
CoreAudio	Cocoa	not yet

Table 3: FAUST applications architectures

Name	System
ladspa	LADSPA plugins
csound	CSOUND opcodes
csounddouble	double precision CSOUND opcodes
maxmsp	Max/MSP externals
vst	native VST plugins
w32vst	windows VST plugins
supercollider	Supercollider plugins
puredata	Puredata externals
Q	Q plugins
Pure	Pure plugins

Table 4: FAUST plugins architectures

A GRAMMAR FOR ANALYZING AND OPTIMIZING AUDIO GRAPHS

Vesa Norilo

Centre for Music & Technology
Sibelius Academy
Helsinki, Finland
vnorilo@siba.fi

ABSTRACT

This paper presents a formal grammar for discussing data flows and dependencies in audio processing graphs. A graph is a highly general representation of an algorithm, applicable to most DSP processes.

To demonstrate and exercise the grammar, three central problems in audio graph processing are examined. The grammar is used to exhaustively analyze the problem of scheduling processing nodes of the graph, examine automatic parallelization as well as signal rate inferral.

The grammar is presented in terms of mathematical set theory, independent of and thus applicable to any conceivable software platform.

1. INTRODUCTION

Most signal processing algorithms are extremely well suited to be represented by graphs, connected networks of nodes. The nodes in the network correspond to processing operations, while the interconnections denote signal flow.

In addition, audio graphs are typically directional. Signal flows traverse the graph from initial sources to eventual destinations, entering processing nodes via inputs and exiting them from their outputs.

Considering how the graph metaphor is so general and widely applicable, it would be highly beneficial if a formal language could be used to reason about graphs in the context of analyzing and transforming audio algorithms.

This paper employs the elementary principles of mathematical set theory in discussing, analyzing and transforming audio graphs. Some straightforward additional notation is introduced to simplify the discussion about data dependencies and node reachability.

As performance is typically critical in audio applications, an immediate field of interest is using the emerging theoretical identities to optimize the computation of audio graphs. The emerging logical language is employed to present and discuss proofs about scheduling and transforming audio graphs, without tying the results to a particular platform or system. The research has been applied to the foundations of the author's work with signal processing compilers[1].

The rest of this paper is organized as follows. First, in Section 2, *Notation*, elementary operators for describing subgraphs and supergraphs are introduced. In Section 3, *Scheduling a DAG*, execution schedule constraints for an audio graph are formally laid out. Section 4 *Parallelization*, discusses rules for automatic parallelization of an audio graph. Section 5, *Signal Rate Optimization* examines how graphs can be analyzed for required update rates. Finally,

Section 6, *Conclusions*, summarizes the paper and the grammar introduced in it.

2. NOTATION

The study of graphs is a relatively recent but growing topic in mathematics. The type of graph best suited for digital computation of audio signals is the directed acyclic graph or DAG[2].

DAGs incorporate the direction of signal flow, so that outputs are fed into inputs, and prohibit cycles in the graph – necessary for a graph to be finitely computable.

Let us begin by defining set-theoretic operators and concepts to enable the analysis of DAGs. For an overview of elementary set theory, the reader is referred to literature[3].

2.1. Reachability

Reachability between two nodes is an intuitive concept. If there exists a path between the nodes, from node to node via connections, the nodes reachable from each other. Let this condition be represented by the general reachability operator, $a \uparrow b$, that produces a boolean truth value.

2.1.1. Upstream Locator

Reachability becomes more useful if the path is constrained. Let us define a more specific reachability operator, the *upstream locator*. Stated as $a \uparrow b$, the operator is true if a can be reached from b by traversing the graph *upstream* – the direction opposite to the signal flow.

2.1.2. Downstream Locator

The inverse of the *upstream locator* is the downstream locator. Used for convenience, the operator can be defined simply as

$$a \downarrow b = b \uparrow a \quad (1)$$

2.2. Subgraphs and Supergraphs

In general graph theory, a subgraph is a set of nodes and interconnections that can be obtained from a larger graph by severing some of the connections. In this paper, we shall adopt a narrower definition of a subgraph.

Let us define a subgraph in terms of data dependency; let a subgraph consist of a particular *root node*, and any nodes reachable from it by traversing the DAG upstream. In other words, the

subgraph is the portion of the DAG through which signal must pass before reaching the input of its root node.

Let a supergraph be the opposite of subgraph. Let the supergraph consist of a root node and all the nodes that can be reached from it by traversing the DAG downstream.

Let a be an arbitrary node in a DAG. Using the locator operators, the subgraph of a can be defined as a section of the universal set \mathbb{U}

$$\uparrow a = a \cup \{x \in \mathbb{U} : x \uparrow a\} \quad (2)$$

Likewise, the supergraph of a is

$$\downarrow a = a \cup \{x \in \mathbb{U} : x \downarrow a\} \quad (3)$$

Supergraphs and subgraphs have an inverse relation;

$$b \in \uparrow a \iff a \in \downarrow b \quad (4)$$

Nested subgraphs and supergraphs imply subsets and supersets;

$$b \in \uparrow a \iff \uparrow a \supseteq \uparrow b \quad (5)$$

$$b \in \downarrow a \iff \downarrow a \supseteq \downarrow b \quad (6)$$

2.3. Summary of Notation

Notation	Meaning
$a \uparrow b$	Node a can be reached from b by traversing the graph upstream
$a \downarrow b$	Node a can be reached from b by traversing the graph downstream
$\uparrow a$	The set of a and all nodes x for which $x \uparrow a$ holds
$\downarrow a$	The set of a and all nodes x for which $x \downarrow a$ holds

3. SCHEDULING A DAG

Let \mathbb{G} be a DAG describing an audio algorithm, consisting of several independent processing nodes. Should this DAG be transformed into a computer program, the first requirement would be to produce a correct processing order for the nodes.

The fundamental scheduling constraint is that the processing of a node can not commence before all its inputs are ready for processing. Let us define two informal operators, *Ready*, indicating whether a node can be processed or not, and *Finished*, indicating whether the node has been processed already. This results in;

$$Ready(a) = \neg(\exists x \in (\uparrow a \setminus \{a\}), \neg Finished(x)) \quad (7)$$

Note that nodes with no input connections, ie. $\uparrow a = \{a\}$ are always ready as they have no dependencies.

A linear list could be constructed from the nodes in graph \mathbb{G} by sorting them according to reachability. A sorting algorithm that operates with a binary less-than predicate could be employed. By utilizing the upstream locator operator as the less-than comparison, a correct schedule can be constructed;

$$a < b \iff a \uparrow b \quad (8)$$

Such sorting algorithms are available in most programming languages, including the standard template library for C++[4]. Simply iterating through such a sorted list is guaranteed to process all the nodes in correct order, provided the sorting algorithm is compatible. This point is expanded in the following subsection.

3.1. Ordering and Reachability

It could be tempting to extend the semantic equivalence of the upstream locator to the full trichotomy of comparison operators;

$$a < b \iff a \uparrow b \quad (9)$$

$$a > b \iff a \downarrow b \quad (10)$$

$$a = b \iff \neg(a \uparrow b \vee a \downarrow b) \quad (11)$$

However, the metaphor breaks down at equality. Consider:

$$\begin{aligned} a &\in \uparrow b \\ c &\notin \uparrow b \\ c &\notin \downarrow b \end{aligned} \quad (12)$$

This would give $a = c, b = c$ but also $a \neq b$, thus the hypothetical equality operator doesn't function as expected.

For less-than predicate sorting to work, the sorting algorithm must not rely on equality derived as in equation 11. The class of acceptable algorithms perform *strict-weak ordering*[5], which relies on a binary less-than operator.

As there may be more than one correct order for any strict-weakly ordered set, the exact result will depend on the sorting algorithm.

4. PARALLELIZATION

As stated in Section 3, there are typically several valid processing schedules for a DAG. In such cases, the ambiguity results from the fact that there are operations that are independent of each other. These operations can be performed in any order or even concurrently.

Any nodes that can be parallelized must therefore be ambiguously ordered. In other words, the nodes should satisfy strict-weak ordering according to upstream reachability in either order. Otherwise, one of the nodes is upstream reachable from the other, and the nodes must be serially processed to honor all the dependencies.

The condition for parallelization can thus be formally stated;

$$\begin{aligned} Parallelizable(a, b) &\iff \neg(a \uparrow b \vee a \downarrow b) \\ &\iff \neg(a \uparrow b \vee b \uparrow a) \\ &\iff \neg(b \in \uparrow a \vee a \in \uparrow b) \end{aligned} \quad (13)$$

More generally, two entire subgraphs are parallelizable if their intersection is the null set;

$$\uparrow a \cap \uparrow b = \emptyset \quad (14)$$

If this condition is met, the subgraphs can be processed independently from each other. Note that this only applies to the narrow definition of a subgraph as defined in Section 2.2, not the subgraphs of general graph theory.

If the intersection yields a non-empty set, the parallelizable components are the relative complements of the subgraphs and their intersection;

$$C = \uparrow a \cap \uparrow b \quad (15)$$

$$A' = \uparrow a \setminus C \quad (16)$$

$$B' = \uparrow b \setminus C \quad (17)$$

$$(18)$$

Therefore, A' and B' can be executed in parallel, but C must be executed before either of them.

4.1. An Algorithm for Parallelization

As there is significant scheduling overhead in parallel computation, it is typically ideal to parallelize as little as possible while still maintaining full utilization of computing resources. A well known method for balancing utilization and overhead is the data flow work queue, where a central pool of available tasks is maintained. Each worker thread pulls a task from the repository, completes it and places any newly available tasks into the pool. Tasks become available as all their inputs are finished, as shown in equation 7.

Inadequate load balancing may cause performance degradation in the case of the data flow work queue. This means that some computational cores are performing useful work while some are not, possibly waiting for tasks that depend on the ones currently being processed. In the case of a general audio DAG, load balancing can be improved by increasing work item granularity – in other words, including fewer processing nodes in each work item. This in turn can cause the scheduling overhead from the growing number of work items to eradicate any gains made from improved load balancing.

Utilizing the results shown above, an algorithm can be constructed that automatically generates a parallelized work schedule. A work item size parameter is introduced to allow for fine tuning the tradeoff between load balancing and scheduling overhead. This algorithm carries the assumption that the computational time consumed by processing a set of nodes can be approximated or measured. The *size* of the work item corresponding to that set is proportional to the computational time.

- Start parallelization from a root node R .
- Obtain the subgraphs of all nodes connected to the inputs of R . Let these be $\{\mathbb{P}_1, \mathbb{P}_2, \dots, \mathbb{P}_n\}$. Let \mathbb{I} be the index set $\{x \in \mathbb{Z} : 0 < x \leq n\}$.
- The serial dependency is

$$\mathbb{S} = \bigcup_{i \in \mathbb{I}} \left(\mathbb{P}_i \cap \bigcup_{j \in \mathbb{I}, j \neq i} \mathbb{P}_j \right) \quad (19)$$

- The parallelizable portions of the DAG are $\{\mathbb{P}_1 \setminus \mathbb{S}, \mathbb{P}_2 \setminus \mathbb{S}, \dots, \mathbb{P}_n \setminus \mathbb{S}\}$
- The parallelizable portions exceeding the size of the chosen work item size threshold are kept. Those that fall below the threshold should be combined, largest with the smallest, until there are no more work items below the threshold or just one item is left.
- For any set \mathbb{P}_k for which $\mathbb{P}_k \cap \mathbb{S} \neq \emptyset$, the respective parallelizable portion $\mathbb{P}_k \setminus \mathbb{S}$ has a dependency on \mathbb{S} , and must be scheduled only after \mathbb{S} is entirely completed.

- If the serial dependency \mathbb{S} exceeds the work item size, it should be recursively parallelized. Let the set of nodes $\{x \in \mathbb{S} : (\downarrow x) \cap \mathbb{S} = \emptyset\}$ form the root set from which a new set of \mathbb{P} subgraphs be built. \mathbb{S} should be considered completed only when the newly parallelized tasks are all completed.

5. SIGNAL RATE OPTIMIZATION

Whereas parallelization is more concerned about when a node set *can* be processed while maintaining data flow integrity, signal rate optimization is about deducing when it *must* be processed.

Not all signals need equally frequent updates, and often significant efficiency can be gained by updating certain node sets at a lower rate. This optimization technique has a strong tradition, with the concept of *control rate* being central in many music software environments ever since the venerable CSound[6].

Another – arguably more desirable – approach is to analyze the signal paths in the audio DAG and automatically determine the desired signal rates for the most typical scenario. This approach, presented here in the terms of the set-theoretic approach of this paper, has previously been described by the author[7].

The automatic process can be guided by inserting non-processing nodes into the DAG whose sole purpose is to guide the signal rate inferral. Once the inferral is completed, these nodes can be removed from the DAG to avoid any overhead.

There are two kinds of sources, streaming and event-based. A streaming source will emit a sampled signal with regular sample intervals. Event-based sources react to some external or internally derived stimulus, producing an update upon receiving, for example, a MIDI event.

In both cases, it is desirable to process the supergraph of a source node according to its update rate. A filter processing the output of an oscillator should work at the same signal rate. Likewise, if an event-driven signal like an user interface slider seldom changes, computations that depend on it should be avoided when unnecessary.

To infer the DAG signal rates, signal sources must be identified. In an audio DAG, these sources are oscillators, audio file players, external audio inputs, user interface control signals and other inputs such as MIDI or OSC[8].

5.1. Source Discovery

A first step in the analysis of the required signal rate for a particular node is to discover which source nodes have the node in their supergraphs. According to equation 3, this can be determined by collecting the source nodes from the subgraph of the node. Let SRC be the set of all source nodes. As a starting point, we could assume that the node needs to be recomputed whenever one of its sources gets updated.

$$\text{Sources}(a) = \uparrow a \cap \text{SRC} \quad (20)$$

5.2. Source Arbitration

There is, however, a further consideration. Stateful processes such as filters or delay lines should be updated only according to their audio signal inputs. This ensures a steady sample rate which would otherwise be compromised by additional updates forced by control signals. Therefore, if a filter node has both an audio input and a

user interface slider in its supergraph, it should ignore the updates by the slider and only update according to the audio rate.

This can be solved by introducing *source priorities*. By providing strict weak ordering[5] of the sources, the desired behavior can be attained. If the audio input has a higher priority than the user interface element, nodes that have both sources should just ignore the user interface updates until the next audio-driven update happens.

5.3. Update Regions

After arbitration, the nodes should be classified according to the arbitrated sources driving them. In fact, both arbitration and classification can be described by simple equations. Let S_{audio} be a high priority source, followed by a medium priority S_{OSC} and a low priority S_{UI} . These correspond to audio, OSC-event and user interface update priorities. Let the node sets they drive be \mathbb{G}_{audio} , \mathbb{G}_{OSC} and \mathbb{G}_{UI} . This gives;

$$\mathbb{G}_{audio} = \Downarrow S_{audio} \quad (21)$$

$$\mathbb{G}_{OSC} = \Downarrow S_{OSC} \setminus \mathbb{G}_{audio} \quad (22)$$

$$\mathbb{G}_{UI} = \Downarrow S_{UI} \setminus (\mathbb{G}_{audio} \cup \mathbb{G}_{OSC}) \quad (23)$$

This list could further be expanded on the simple principle that each source drives all the nodes in its supergraph, except those that also belong to a supergraph of a higher priority source.

Such a system of graphs can be scheduled easily by processing the node sets in reverse order of priority. From equations 4 and 23 it can be deduced that;

$$\mathbb{G}_{audio} \cup \mathbb{G}_{OSC} = \Downarrow S_{audio} \cup \Downarrow S_{OSC} \quad (24)$$

$$\mathbb{G}_{UI} \cap \Downarrow S_{audio} = \emptyset \quad (25)$$

$$\mathbb{G}_{UI} \cap \Downarrow S_{OSC} = \emptyset \quad (26)$$

$$\forall x \in \mathbb{G}_{UI}, (\uparrow x) \cap \mathbb{G}_{OSC} = \emptyset \quad (27)$$

$$\forall x \in \mathbb{G}_{UI}, (\uparrow x) \cap \mathbb{G}_{audio} = \emptyset \quad (28)$$

Thus, no subgraph of any node in \mathbb{G}_{UI} can contain any nodes that also belong to \mathbb{G}_{OSC} or \mathbb{G}_{audio} . Therefore, \mathbb{G}_{UI} can safely be scheduled before the higher priority blocks. The proof can be extended to show that any lower priority node group can always be safely scheduled before higher priority node groups. This is especially important in the case where both sources are driven from a coherent clock source, such as traditional audio and control signals. Failure to schedule coherent clock sources according to their priority would result in potential undesired delays at signal rate boundaries.

5.4. Priority Inversal

In many algorithms, more precise control of signal rates is required. With just the inferral system described so far would make it impossible, for example, to derive MIDI events from audio signals at any rate below the audio sampling rate. To solve this problem, it is necessary to be able to override the source priorities locally, at a specific DAG junction.

The best possible priority inversal mechanism is still a topic of active research. As an initial solution, priority escalation is suggested. A special source could be generated for any DAG junctions where priority inversal is desired. This source would have a higher

priority than the one it is meant to override. Scheduling-wise, this additional source should be processed according to the reverse priority order, generating some additional bookkeeping overhead.

6. CONCLUSIONS

In this paper, elementary concepts for formally discussing directed acyclic graphs in audio context were introduced. These concepts include the upstream and downstream reachability operators, as well as the construction of subgraphs and supergraphs. Taken together, these devices facilitate set-theoretic discussion of processing directed acyclic graphs for audio signals.

Three practical, highly important problems were examined using the newfound grammar. The problem of scheduling operations described as a signal processing graphs was examined and deemed a strict-weak order based on a simple reachability operator. The ambiguity of that strict-weak order was leveraged to analyze the problem of concurrently executing portions of an audio processing graph. Finally, the grammar was utilized to discuss automatic signal rate optimization and discover the additional scheduling constraints such a system imposes.

The concepts form the basis of the author's work on signal processing languages and compilers[1]. They are presented here independently from any programming language or system, instead employing the notation of mathematical set theory. The concepts are not overwhelmingly difficult, but utilization of formal grammar helps avoid ambiguously worded statements and pseudo-rules. Further, statements in a formal language lend themselves to further reasoning, identities and proofs. This is of vital importance when constructing compilers and interpreters, especially in the case of automatic parallelization of user algorithms. This paper is written in the hopes of providing assistance in the form of a grammar to the researchers working with these problems.

7. REFERENCES

- [1] Vesa Norilo, "Introducing Kronos - A Novel Approach to Signal Processing Languages," in *Proceedings of the Linux Audio Conference*, Frank Neumann and Victor Lazzarini, Eds., Maynooth, Ireland, 2011, pp. 9–16, NUIM.
- [2] F Harary, Robert Z Norman, and D Cartwright, *Structural models: An introduction to the theory of directed graphs*, Wiley, 1965.
- [3] H B Enderton, "The Joy of Sets. Fundamentals of Contemporary Set Theory.," *The Journal of Symbolic Logic*, vol. 59, no. 4, pp. 1441, 1994.
- [4] Alexander Stepanov and Meng Lee, *The Standard Template Library*, Number X3J16/94-0095, WG21/N0482. Prentice-Hall, 1995.
- [5] Bernd Schröder, *Ordered Sets: An Introduction*, Birkhäuser Boston, 2002.
- [6] Richard Boulanger, *The Csound Book*, vol. 309, MIT Press, 2000.
- [7] Vesa Norilo and Mikael Laurson, "Unified Model for Audio and Control Signals," in *Proceedings of ICMC*, Belfast, Northern Ireland, 2008.
- [8] Matthew Wright, Adrian Freed, and Ali Momeni, "Open-Sound Control: State of the Art 2003," *Time*, pp. 153–159, 2003.

STATE OF THE ART IN SOUND TEXTURE SYNTHESIS

Diemo Schwarz

Real-Time Music Interaction Team
IRCAM–CNRS–UPMC, UMR STMS

Paris, France

diemo.schwarz@ircam.fr

ABSTRACT

The synthesis of sound textures, such as rain, wind, or crowds, is an important application for cinema, multimedia creation, games and installations. However, despite the clearly defined requirements of naturalness and flexibility, no automatic method has yet found widespread use. After clarifying the definition, terminology, and usages of sound texture synthesis, we will give an overview of the many existing methods and approaches, and the few available software implementations, and classify them by the synthesis model they are based on, such as subtractive or additive synthesis, granular synthesis, corpus-based concatenative synthesis, wavelets, or physical modeling. Additionally, an overview is given over analysis methods used for sound texture synthesis, such as segmentation, statistical modeling, timbral analysis, and modeling of transitions.

1. INTRODUCTION

The synthesis of sound textures is an important application for cinema, multimedia creation, games and installations. Sound textures are generally understood as sound that is composed of many micro-events, but whose features are stable on a larger time-scale, such as rain, fire, wind, water, traffic noise, or crowd sounds. We must distinguish this from the notion of *soundscape*, which describes the sum of sounds that compose a scene, some components of which could be sound textures.

There are a plethora of methods for sound texture synthesis based on very different approaches that we will try to classify in this state-of-the-art article. We'll start by a definition of the terminology and usages (sections 1.1– 1.3), before giving an overview of the existing methods for synthesis and analysis of sound textures (sections 2 and 3), and some links to the first available software products (section 4). Finally, the discussion (section 5) and conclusion (section 6) also point out some especially noteworthy articles that represent the current state of the art.

1.1. Definition of Sound Texture

An early thorough definition, and experiments on the perception and generation of sound textures were given by Saint-Arnaud [74] and Saint-Arnaud and Popat [75], summarised in the following visual analogy:

A sound texture is like wallpaper: it can have local structure and randomness, but the characteristics of the fine structure must remain constant on the large scale.

Figure 1 illustrates this statement. They culminate in the following working definition:

1. Sound textures are formed of basic sound elements, or atoms;
2. atoms occur according to a higher-level pattern, which can be periodic, random, or both;
3. the high-level characteristics must remain the same over long time periods (which implies that there can be no complex message);
4. the high-level pattern must be completely exposed within a few seconds (“attention span”);
5. high-level randomness is also acceptable, as long as there are enough occurrences within the attention span to make a good example of the random properties.

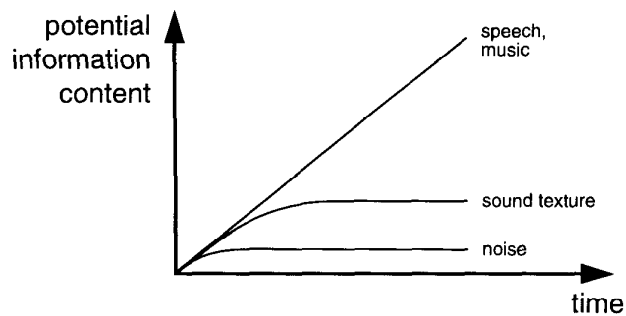


Figure 1: Potential information content of a sound texture vs. time (from Saint-Arnaud and Popat [75]).

1.1.1. What Sound Texture is Not

Attempting a negative definition might help to clarify the concept. We exclude from sound textures the following:

Contact sounds from interaction with objects, such as impact, friction, rolling sounds, treated in many works close to sound texture synthesis [1, 15, 48, 65, 87]. These sounds violate the “wallpaper” property.

Sound scapes are often treated together with sound textures, since they always contain sound textures. However, sound scapes also comprise information-rich event-type sounds, as further explained in section 1.1.2.



Figure 2: Examples of natural and synthesised oriented oscillating patterns from Peyré [70].

Sound design is the wider context of creating interaction sounds, sound scapes, and sound textures. Literature in the field often contains useful methods for sound texture design [10, 14, 58–61].

In some cases of music composition or performance, *sound texture* is used to mean *non-tonal*, *non-percussive* sound material, or *non-harmonic*, *non-rhythmic* musical material.

See also Strobl [84] for an investigation of the term *texture* outside of sound, such as in textiles, typography, gastronomy.

1.1.2. Sound Scapes

Because sound textures constitute a vital part of sound scapes, it is useful to present here a very brief introduction to the classification and automatic generation of *sound scapes*. Also, the literature about sound scapes is inevitably concerned about the synthesis and organisation of sound textures.

The first attempts at definition and classification of sound scapes have been by Murray Schafer [76], who distinguishes *keynote*, *signal*, and *soundmark* layers in a soundscape, and proposes a referential taxonomy incorporating socio-cultural attributes and ecological acoustics.

Gaver [36], coming from the point of view of acoustic ecology, organises sounds according to their physical attributes and interaction of materials.

Current work related to sound scapes are frequent [8, 9, 33, 58–61, 88, 89].

1.2. Existing Attempts at Classification of Texture Synthesis

As a starting point, Strobl et al. [85] provide an attempt at a definition of sound texture, and an overview of work until 2006. They divide the reviewed methods into two groups:

Methods from computer graphics Transfer of computer graphics methods for visual texture synthesis applied to sound synthesis [22, 64, 67]. See figure 2 for examples of textured images.

Methods from computer music Synthesis methods from computer music or speech synthesis applied to sound texture synthesis [4, 7, 17, 42, 43, 94].

A newer survey of tools in the larger field of sound design and composition [58] propose the same classification by synthesis method as elaborated in section 2 below. The article makes a point that different classes of sound require different tools (“*A full toolbox means the whole world need not look like a nail!*”) and gives a list of possible matches between different types of sound and the sound synthesis methods on which they work well.

In an article by Filatriau and Arfib [31], texture synthesis algorithm are reviewed from the point of view of gesture-controlled instruments, which makes it worthwhile to point out the different usage contexts of sound textures in the following section.

1.3. Different Usages and Significations

It is important to note that there is a possible confusion in the literature about the precise signification of the term *sound texture* that is dependent on the intended usage. We can distinguish two frequently occurring usages:

Expressive texture synthesis Here, the aim is to interactively generate sound for music composition, performance, or sound art, very often as an expressive digital musical instrument (DMI). *Sound texture* is then often meant to distinguish the generated sound material from tonal and percussive sound, i.e. sound texture is anything that is predominantly defined by timbre rather than by pitch or rhythm.

The methods employed for expressive texture generation can give rise to naturally sounding textures, as noted by

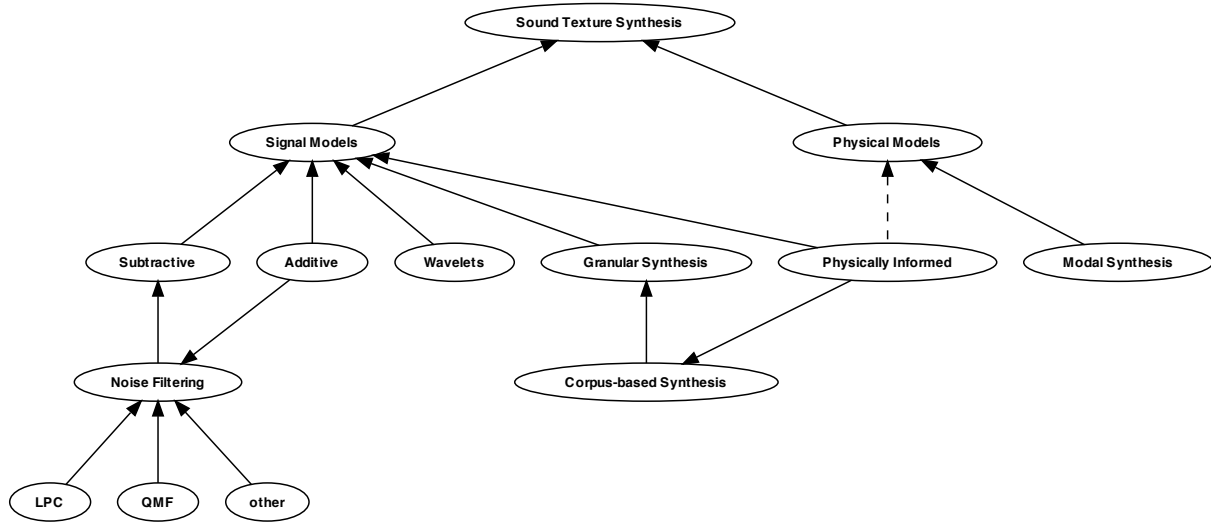


Figure 3: Classification hierarchy of sound texture synthesis methods. Dashed arrows represent use of information.

Di Scipio [17], but no systematic research on the usable parameter space has been carried out, and it is up to the user (or player) to constrain herself to the natural sounding part. This strand of sound texture is pursued in the already mentioned review in [31], and their follow-up work [32].

Natural texture resynthesis tries to synthesise environmental or human textural sound as part of a larger soundscape, amongst others for audio–visual creation like cinema or games. Often, a certain degree of realism is striven for (like in photorealistic texture image rendering), but for most applications, either symbolic or impressionistic *credible texture synthesis* is actually sufficient, in that the textures convey the desired ambience or information, e.g. in simulations for urbanistic planning. All but a few examples of the work described in the present article is aimed this usage.

2. CLASSIFICATION OF SYNTHESIS METHODS

In this section, we will propose a classification of the existing methods of sound texture synthesis. It seems most appropriate to divide the different approaches by the synthesis methods (and analysis methods, if applicable) they employ:

- Noise filtering (section 2.1) and additive sinusoidal synthesis (section 2.2)
- Physical modeling (section 2.3) and physically-informed signal models
- Wavelet representation and resynthesis (section 2.4)
- Granular synthesis (section 2.5) and its content-based extension corpus-based concatenative synthesis (section 2.6)
- Non-standard synthesis methods, such as fractal or chaotic maps (section 2.7)

Figure 3 gives an overview over the classes and their relationships. Other possible aspects for classification are the degree of dependency on a model, the degree to which the method is data-driven, the real-time capabilities, and if the method has been formally evaluated in listening tests. Some of these aspects will be discussed in section 5.

2.1. Subtractive Synthesis

Noise filtering is the “classic” synthesis method for sound textures, often based on specific modeling of the source sounds.

Based on their working definition listed in section 1.1, Saint-Arnaud and Popat [75] build one of the first analysis–synthesis models for texture synthesis, based on 6-band Quadrature Mirror filtered noise.

Athineos and Ellis [4] and Zhu and Wyse [94] apply cascaded time and frequency domain linear prediction (CTFLP) analysis and resynthesis by noise filtering. The latter resynthesise the background din and the previously detect foreground events (see section 3.1.3) by applying the time and frequency domain LPC coefficients to noise frames with subsequent overlap–add synthesis. The events are sequenced by a Poisson distribution. The focus here is data reduction for transmission by low-bitrate coding.

McDermott et al. [53] apply statistical analysis (see section 3.2) to noise filtering synthesis, restrained to unpitched static textures like rain, fire, water only.

Peltola et al. [69] synthesises different characters of hand-clapping sounds by filters tuned to recordings of claps and combines them into a crowd by statistical modeling of different levels of enthusiasm and flocking behaviour of the crowd.

The venerable collection by Farnell [30], also available online¹, gives many sound and PUREDATA patch examples for synthesising

¹http://obiwannabe.co.uk/tutorials/html/tutorials_main.html

various sound textures and sound effects by oscillators and filters, carefully tuned according to insights into the phenomenon to be simulated, as in this quote about rain:

"What is the nature of rain? What does it do?" According to the lyrics of certain shoe-gazing philosophies it's "Always falling on me", but that is quite unhelpful. Instead consider that it is nearly spherical particles of water of approximately 1-3mm in diameter moving at constant velocity impacting with materials unknown at a typical flux of 200 per second per meter squared. All raindrops have already attained terminal velocity, so there are no fast or slow ones. All raindrops are roughly the same size, a factor determined by their formation at precipitation under nominally uniform conditions, so there are no big or small raindrops to speak of. Finally raindrops are not "tear" shaped as is commonly held, they are in fact near perfect spheres. The factor which prevents rain being a uniform sound and gives rain its diverse range of pitches and impact noises is what it hits. Sometimes it falls on leaves, sometimes on the pavement, or on the tin roof, or into a puddle of rainwater.

2.2. Additive Sinusoidal + Noise Synthesis

Filtered noise is often complemented by oscillators in the additive sinusoidal partials synthesis method.

In the QCITY project², the non-real time simulation of traffic noise is based on a sinusoids+noise sound representation, calibrated according to measurements of motor states, exhaust pipe type, damping effects. It allows to simulate different traffic densities, speeds, types of vehicles, tarmacs, damping walls, etc. [38]. The calculation of sound examples can take hours.

Verron [93] proposes in his PhD thesis and in other publications [91, 92], 7 physically-informed models from Gaver's [36] 3 larger classes of environmental sounds: liquids, solids, aerodynamic sounds. The models for impacting solids, wind, gushes, fire, water drops, rain, and waves are based on 5 empirically defined and parameterised sound atoms: modal impact, noise impact, chirp impact, narrow band noise, wide band noise. Each model has 2–4 low-level parameters (with the exception of 32 band amplitudes for wide band noise).

Verron then painstakingly maps high-level control parameters like *wind force and coldness*, *rain intensity*, *ocean wave size* to the low-level atom parameters and density distribution.

The synthesis uses the FFT⁻¹ method [73] that is extended to include spatial encoding into the construction of the FFT, and then one IFFT stage per output channel.³

2.3. Physical Modeling

Physical modeling can be applied to sound texture synthesis, with the drawback that a model must be specifically developed for each

class of sounds to synthesise (e.g. friction, rolling, machine noises, bubbles, aerodynamic sounds) [63, 64], the latter adding an extraction of the impact impulse sound and a perceptual evaluation of the realism of synthesised rolling sounds (see also Lagrange et al. [47]). Often, modal resonance models are used [90], where the in-expensively synthesisable modes are precalculated from expensive rigid body simulations.

Other signal-based synthesis methods are often *physically-informed* [12, 13] in that they control signal models by the output of a physical model that captures the behaviour of the sound source. See, e.g. Cook [14], Verron [93] (also described in section 2.2), Picard et al. [71], or the comprehensive toolbox by Menzies [55] (see section 4 for its implementation).

The synthesis of liquid sounds described by Doel [20] is a combination of a physically informed sinusoidal signal model for single bubble sounds (going back to [51]), and an empirical phenomenological model for bubble statistics, resulting in a great sound variety ranging from drops, rain, air bubbles in water to streams and torrents.

An extreme example is the synthesis of sounds of liquids by fluid simulations [62], deriving sound control information from the spherical harmonics of individually simulated bubbles (up to 15000).⁴

2.4. Wavelets

The multiscale decomposition of a signal into a wavelet coefficient tree has been first applied to sound texture synthesis by El-Yaniv et al. [26] and Dubnov et al. [22], and been reconsidered by Kersten and Purwins [45].⁵

Here, the multiscale wavelet tree signal and structure representation is resampled by reorganising the order of paths down the tree structure. Each path then resynthesises a short bit of signal by the inverse wavelet transform.

These approaches take inspiration from image texture analysis and synthesis and try to model temporal dependencies as well as hierarchical dependencies between different levels of the multi-level tree representation they use. Kersten and Purwins's work is in an early stage where the overall sound of the textures is recognisable (as shown by a quantitative evaluation experiment), but the resulting structure seems too fine-grained, because the sequence constraints of the original textures are actually not modeled, such that the fine temporal structure gets lost. This violates the *autocorrelation feature* found important for audio and image textures by McDermott et al. [53] and Fan and Xia [29].

An model-based approach using wavelets for modeling stochastic-based sounds is pursued by Miner and Caudell [56]. Parameterizations of the wavelet models yield a variety of related sounds from a small set of dynamic models.

Another wavelet-based approach is by Kokaram and O'Regan [46, 66], based on Efros and Leung's algorithm for image texture synthesis [23]. Their multichannel synthesis achieves a large segment size well adapted to the source (words, baby cries, gear shifts,

²<http://qcit.eu/dissemination.html>

³Binaural sound examples (sometimes slightly artificial sounding) and one video illustrating the high-level control parameters and the difference between point and extended spatial sources can be found on <http://www.charlesverron.com/thesis/>.

⁴<http://gamma.cs.unc.edu/SoundingLiquids>

⁵Sound examples are available at <http://mtg.upf.edu/people/skersten?p=Sound%20Texture%20Modeling>

drum beats) and thus a convincing and mostly artefact-free resynthesis.⁶

2.5. Granular Synthesis

Granular synthesis uses snippets of an original recording, and possibly a statistical model of the (re)composition of the grains [7, 22, 26, 35, 42, 43, 67]. The optimal grain size is dependent on the typical time-scale of the texture. If chosen sufficiently long, the short-term micro-event distribution is preserved within a grain, while still allowing to create a non-repetitive long-term structure.

Lu et al. [50] recombine short segments, possibly with transposition, according to a model of transition probabilities (see section 3.4). They explicitly forbid short backward transitions to avoid repetition. The segmentation is based on a *novelty score* on MFCCs, in the form of a similarity matrix.

Strobl [84] studies the methods by Hoskinson and Pai [42, 43] and Lu et al. [50] in great detail, improves the parameters and resynthesis to obtain “perceptually perfect segments of input textures”, and tries a hybridisation between them [84, chapter 4]. She then implements Lu et al.’s method in an interactive real-time PUREDATA patch.

2.6. Corpus-based Synthesis

Corpus-based concatenative synthesis can be seen as a content-based extension of granular synthesis [78, 79]. It is a new approach to sound texture synthesis [11, 80, 82, 83]. Corpus-based concatenative synthesis makes it possible to create sound by selecting snippets of a large database of pre-recorded audio (the corpus) by navigating through a space where each snippet is placed according to its sonic character in terms of audio descriptors, which are characteristics extracted from the source sounds such as pitch, loudness, and brilliance, or higher level meta-data attributed to them. This allows one to explore a corpus of sounds interactively or by composing paths in the space, and to create novel timbral evolutions while keeping the fine details of the original sound, which is especially important for convincing sound textures.

Finney [33] uses a corpus of unstructured recordings from the *free-sound* collaborative sound database⁷ as base material for sample-based sound events and background textures in a comprehensive sound scape synthesis application (see section 4, also for evaluation by a subjective listening test). The recordings for sound texture synthesis are segmented by MFCC+BIC (see section 3.1.2) and high- and low-pass filtered to their typical frequency ranges. Spectral outliers (outside one standard deviation unit around the mean MFCC) are removed. Synthesis then chooses randomly out of a cluster of the 5 segments the MFCCs of which are closest. How the cluster is chosen is not explicitly stated.

Schwarz and Schnell [80] observe that existing methods for sound texture synthesis are often concerned with the extension of a given recording, while keeping its overall properties and avoiding artefacts. However, they generally lack controllability of the resulting sound texture. They propose two corpus-based methods of statistical modeling of the audio descriptor distribution of texture recordings using histograms and Gaussian mixture models. The models

can be interpolated to steer the evolution of the sound texture between different target recordings (e.g. from light to heavy rain). Target descriptor values are stochastically drawn from the statistical models by inverse transform sampling to control corpus-based concatenative synthesis for the final sound generation, that can also be controlled interactively by navigation through the descriptor space.⁸ See also section 4 for the freely available CATART application that served as testbed for interactive sound texture synthesis. To better cover the target descriptor space, they expand the corpus by automatically generating variants of the source sounds with transformations applied, and storing only the resulting descriptors and the transformation parameters in the corpus. A first attempt of perceptual validation of the used descriptors for wind, rain, and wave textures has been carried out by subject tests [52], based on studies on the perception of environmental sound [57, 86].

The work by Picard et al. [71] (section 2.3) has a corpus-based aspect in that it uses grain selection driven by a physics engine.

Dobashi et al. [18, 19] employ physically informed corpus-based synthesis for the synthesis of aerodynamic sound such as from wind or swords. They precompute a corpus of the aerodynamic sound emissions of point sources by computationally expensive turbulence simulation for different speeds and angles, and can then interactively generate the sound of a complex moving object by lookup and summation.

2.7. Non-standard Synthesis Methods

Non-standard synthesis methods, such as fractal synthesis or chaotic maps, generated by iterating nonlinear functions, are used most often for expressive texture synthesis [17, 32], especially when controlled by gestural input devices [3, 31].

3. ANALYSIS METHODS FOR SOUND TEXTURES

Methods that analyse the properties of sound textures are concerned with segmentation (section 3.1), the analysis of statistical properties (section 3.2) or timbral qualities (section 3.3), or the modeling of the sound source’s typical state transitions (section 3.4).

3.1. Segmentation of Source Sounds

3.1.1. Onset Detection

O’Modhrain and Essl [65] describe a granular analysis method they call *grainification* of the interaction sounds with actual grains (pebbles in a box, starch in a bag), in order to expressively control granular synthesis (this falls under the use case of expressive texture synthesis in section 1.3): By threshold-based attack detection with a retrigger limit time, they derive the grain attack times, volume (by picking the first peak after the attack), and spectral content (by counting the zero-crossings in a 100 sample window after the attack). These parameters control a granular synthesizer’s trigger, gain and transposition. See also Essl and O’Modhrain [28].

Lee et al. [48] estimate contact events for segmentation of rolling sounds on a high-pass filtered signal, on which an energy threshold

⁶Sound examples are available at http://www.netsoc.tcd.ie/~dee/STS_EUSIPCO.html.

⁷<http://www.freesound.org/>

⁸Sound examples can be heard on http://imtr.ircam.fr/imtr/Sound_Texture_Synthesis.

is applied. Segments are then modeled by LPC filters on several bands for resynthesis.

3.1.2. Spectral Change Detection

Lu et al. [50] segment sounds based on a *novelty score* on MFCCs, in the form of a similarity matrix. This also serves to model transition probabilities (see section 3.4). The analysis has been improved upon by Strobl [84].

Finney [33] (see also sections 2.6 and 4) uses a method of segmenting environmental recordings using the *Bayesian Information Criterion* (BIC) on MFCCs [2], while enforcing a minimum segment length dependent on the type of sounds: The segment length should correspond to the typical event length.

3.1.3. LPC Segmentation

Kauppinen and Roth [44] segment sound into locally stationary frames by LPC pulse segmentation, obtaining the optimal frame length by a stationarity measure from a long vs. short term prediction. The peak threshold is automatically adapted by the median filter of the spectrum derivative.

Similarly, Zhu and Wyse [94] detect foreground events by frequency domain linear predictive coding (FDLPC), which are then removed to leave only the 'din' (the background sound). See also section 2.1 for their corresponding subtractive synthesis method.

3.1.4. Wavelets

Hoskinson and Pai [42, 43] (see also section 2.5) segment the source sounds into *natural grains*, which are defined by the *minima* of the energy changes in the first 6 wavelet bands, i.e. where the sound is the most stable.

3.1.5. Analysis into Atomic Components

Other methods [49, 50, 59, 60, 67] use an analysis of a target sound in terms of event and spectral components for their statistical recombination. They are linked to the modelisation of impact sounds by wavelets by Ahmad et al. [11].

Bascou [5], Bascou and Pottier [6] decompose a sound by *Matching Pursuit* into time-frequency atoms from a dictionary manually built from "characteristic" grains of the sound to decompose.

3.2. Analysis of Statistical Properties

Dubnov et al. [22], El-Yaniv et al. [26] apply El-Yaniv et al.'s [25] Markovian unsupervised clustering algorithm to sound textures, thereby constructing a discrete statistical model of a sequence of paths through a wavelet representation of the signal (see section 2.4).

Zhu and Wyse [94] estimate the density of foreground events, singled out of the texture by LPC segmentation (see section 3.1.3). Masurelle [52] developed a simple density estimation of impact events based on O'Modhrain and Essl [65], applicable e.g. to rain. For the same specific case, Doel [20] cites many works about the statistics of rain.

McDermott et al. [53] (see section 2.4) propose a neurophysically motivated statistical analysis of the kurtosis of energy in subbands, and apply these statistics to noise filtering synthesis (later also applied to classification of environmental sound [27]).

3.2.1. Analysis not for Synthesis

There is work concerned with analysis and classification of sound textures, which is not relevant for synthesis, like Dubnov and Tishby [21], who use higher-order spectra for classification of environmental sounds, or by Desainte-Catherine and Hanna [16], who propose statistical descriptors for noisy sounds.

In the recent work by Grill [37] for an interactive sound installation, a corpus-based synthesis system plays back samples of soundscapes matching the participants' noises. While the synthesis part is very simple, the matching part is noteworthy for its use of *fluctuation patterns*, i.e. the modulation spectrum for all bark bands of a 3 second segment of texture. This 744-element feature vector was then reduced to 24 principal components prior to matching.

3.3. Analysis of Timbral Qualities

Hanna et al. [40] note that, in the MIR domain, there is little work about audio features specifically for noisy sounds. They propose classification into the 4 sub-classes coloured, pseudo-periodic, impulsive noise (rain, applause), and noise with sinusoids (wind, street soundscape, birds). They then detect the transitions between these classes using a Bayesian framework. This work is generalised to a sound representation model based on stochastic sinusoids [39, 41].

Only corpus-based concatenative synthesis methods try to characterise the sonic contents of the source sounds by perceptually meaningful audio descriptors: [24, 78–80, 82, 83]

3.4. Clustering and Modeling of Transitions

Saint-Arnaud [74] builds clusters by k-means of their input sound atoms (filter band amplitudes) using the *Cluster based probability model* [72]. The amplitudes are measured at the current frame and in various places in the past signal (as defined by a *neighbourhood mask*) and thus encode the typical transitions occurring in the sound texture. Saint-Arnaud's master's thesis [74] focuses on classification of sound textures, also with perceptual experiments, while the later article [75] extends the model to analysis by noise filtering (section 2.1).

Lu et al. [50] model transition probabilities based on a similarity matrix on MFCC frames. Hoskinson and Pai [42, 43] also model transitions based on smoothness between their wavelet-segmented *natural grains* (section 3.1.4). Both methods have been studied in detail and improved upon by Strobl [84].

4. AVAILABLE SOFTWARE

Freely or commercially available products for sound textures are very rare, and mostly specific to certain types of textures. The

only commercial product the author is aware of is the crowd simulator CROWD CHAMBER⁹, that takes a given sound file to multiply the sources, probably using PSOLA-based pitch shifting, time-stretching and filtering effects. That means, it is not actually a texture synthesiser, but an effects processor that adds a “crowd” effect on an existing voice recording. The provided example sounds are very unconvincing.

Finney [33] presents a full soundscape synthesis system based on concatenative and sample-based playback (see sections 2.6 and 3.1.2). The synthesis and interaction part is integrated into Google Street View.¹⁰ Notable and related to sound textures is the precise modeling of traffic noise with single samples of passing cars, categorised by car type and speed, that are probabilistically recombined according to time of day, number of street lanes, and with traffic lights simulated by clustering. The evaluation also reported in [34] concentrates on the immersive quality of the generated sound scapes in a subjective listening test with 8 participants. Interestingly, the synthetic sound scapes rate consistently higher than actual recordings of the 6 proposed locations.

The PHYA framework [54, 55] is a toolbox of various physically motivated filter and resonator signal models for impact, collision, and surface sounds.¹¹

The author’s CATART system [83] for interactive real-time corpus-based concatenative synthesis is implemented in MAX/MSP with the extension libraries FTM&Co.¹² and is freely available.¹³ It allows to navigate through a two- or more-dimensional projection of the descriptor space of a corpus of sound segments in real-time using the mouse or other gestural controllers, effectively extending granular synthesis by content-based direct access to specific sound characteristics. This makes it possible to recreate dynamic evolutions of sound textures with precise control over the resulting timbral variations, while keeping the micro-event structure intact, as soon as the segments are long enough, described in section 2.6. One additional transformation is the augmentation of the texture density by triggering at a faster rate than given by the segments’ length, thus layering several units, which works very well for textures like rain, wind, water, or crowds.⁸

The descriptors are calculated within the CATART system by a modular analysis framework [81]. The used descriptors are: fundamental frequency, periodicity, loudness, and a number of spectral descriptors: spectral centroid, sharpness, flatness, high- and mid-frequency energy, high-frequency content, first-order autocorrelation coefficient (expressing spectral tilt), and energy. Details on the descriptors used can be found in [77] and [68].

5. DISCUSSION

Concerning the dependency on a specific model, we can see that the presented methods fall clearly on one of two sides of a strong dichotomy between rule-based and data-driven approaches: The methods using a low-level signal or physical model (sections 2.2–2.3) are almost all based on a very specific modeling of the sound texture generating process, except the first three methods using

noise filtering by statistical modeling. The methods using segments of signal or wavelet coefficients (sections 2.4–2.6) are, by their data-driven nature, more generally applicable to many different texture sounds, and far more independent from a specific texture model.

Also, physical models do not provide a direct link between their internal parameters, and the characteristics of the produced sound. As Menzies [55] notes:

In principle, sound in a virtual environment can be reproduced accurately through detailed physical modelling. Even if this were achieved, it is not enough for the Foley sound designer, who needs to be able to shape the sound according to their own imagination and reference sounds: explicit physical models are often difficult to calibrate to a desired sound behaviour although they are controlled directly by physical parameters.

Physically-informed models allow more of this flexibility but still expose parameters of a synthesis model that might not relate directly to a perceived sound character. What’s more, the physical and signal models’ parameters might capture a certain variety of a simulated sound source, but will arguably be limited to a smaller range of nuances, and include to a lesser extent the context of a sound source, than the methods based on actual recordings (wavelets and corpus-based concatenative synthesis).

5.1. Perception and Interaction

General studies of the perception of environmental sound textures are rare, with the exception of [53, 57, 86], and systematic evaluation of the quality of the synthesised sound textures by formal listening tests is only beginning to be carried out in some of the presented work, e.g. [52, 63]. Only Kokaram and O’Regan [46, 66] have taken the initiative to start defining a common and comparable base of test sounds by adopting the examples from El-Yaniv et al. [26] and Dubnov et al. [22] as test cases.

Finally, this article concentrated mainly on the sound synthesis and analysis models applied to environmental texture synthesis, and less on the way how to control them, or the interactivity they afford. Gestural control seems here a promising approach for interactive generation of sound textures [3, 31, 52].

5.2. Recommended Reading

While this article strove to give a comprehensive overview of existing methods for sound texture synthesis and analysis, some of the work stands out, representing the state of the art in the field:

- Finney [33] for the introduction to sound scapes, the reference to the MFCC+BIC segmentation method, and the precise traffic modeling.
- Verron [93] and Farnell [30] for the detailed account of physically informed environmental sound synthesis that gives an insight about how these sounds work.
- Kokaram and O’Regan [46, 66] and Schwarz and Schnell [80] for the most convincing results so far.

⁹http://www.quikquak.com/Prod_CrowdChamber.html

¹⁰http://dev.mtg.upf.edu/soundscape/media/StreetView/streetViewSoundscaper2_0.html

¹¹ Available at <http://www.zenprobe.com/phya/>.

¹²<http://ftm.ircam.fr>

¹³<http://imtr.ircam.fr/imtr/CataRT>

6. CONCLUSION

We have seen that, despite the clearly defined problem and application context, the last 16 years of research into sound texture synthesis have not yet brought about a prevailing method that satisfies all requirements of realism and flexibility. Indeed, in practice, the former is always the top priority, so that the flexibility of automated synthesis methods is eschewed in favour of manual matching and editing of texture recordings in post-production, or simple triggering of looped samples for interactive applications such as games.

However, the latest state-of-the-art results in wavelet resynthesis [46, 66] and descriptor-based granular synthesis [80] promise practical applicability because of their convincing sound quality.

7. ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers and Stefan Kersten for their careful re-reading, additional references, and pertinent remarks.

The work presented here is partially funded by the *Agence Nationale de la Recherche* within the project *Topophonie*, ANR-09-CORD-022, <http://topophonie.fr>.

8. REFERENCES

- [1] W. Ahmad, H. Hacıhabiboglu, and A.M. Kondoz. Analysis-Synthesis Model for Transient Impact Sounds by Stationary Wavelet Transform and Singular Value Decomposition. In *Proceedings of the International Computer Music Conference (ICMC)*, 2008.
- [2] X Anguera and J Hernando. Xbic: Real-time cross probabilities measure for speaker segmentation, 2005.
- [3] D. Arfib. Gestural strategies for specific filtering processes. *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, 2002. URL http://www2.hsu-hh.de/EWEB/ANT/dafx2002/papers/DAFX02_Arfib_Couturier_Kessous_gestural_strategies.pdf.
- [4] M. Athineos and D.P.W. Ellis. Sound texture modelling with linear prediction in both time and frequency domains. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 5:V-648–51 vol.5, April 2003. ISSN 1520-6149.
- [5] C Bascou. Modélisation de sons bruités par la synthèse granulaire. Rapport de stage de DEA ATIAM, Université Aix-Marseille II, 2004.
- [6] C. Bascou and L. Pottier. New sound decomposition method applied to granular synthesis. In *Proceedings of the International Computer Music Conference (ICMC)*, Barcelona, Spain, 2005.
- [7] C. Bascou and L. Pottier. GMU, a flexible granular synthesis environment in Max/MSP. In *Proceedings of the International Conference on Sound and Music Computing (SMC)*, 2005.
- [8] D. Birchfield, N. Mattar, and H. Sundaram. Design of a generative model for soundscape creation. In *Proceedings of the International Computer Music Conference (ICMC)*, Barcelona, Spain, 2005.
- [9] D. Birchfield, N. Mattar, H. Sundaram, A. Mani, and B. Shevade. Generative Soundscapes for Experiential Communication. *Society for Electro Acoustic Music in the United States (SEAMUS)*, 2005.
- [10] P. Cano, L. Fabig, F. Gouyon, M. Koppenberger, A. Loscos, and A. Barbosa. Semi-automatic ambiance generation. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, Naples, Italy, 2004.
- [11] Marc Cardle. Automated Sound Editing. Technical report, Computer Laboratory, University of Cambridge, UK, May 2004. URL <http://www.cl.cam.ac.uk/users/mpc33/Cardle-Sound-Synthesis-techreport-2004-low-quality.pdf>.
- [12] P Cook. Physically informed sonic modeling (PhISM): Percussive synthesis. *Proceedings of the International Computer Music Conference (ICMC)*, 1996. URL http://scholar.google.com/scholar?hl=en&q=cook+phism&bav=on.2,or.r_gc.r_pw.&biw=1584&bih=783&um=1&ie=UTF-8&sa=N&tab=ws#3.
- [13] PR Cook. Physically informed sonic modeling (phism): Synthesis of percussive sounds. *Computer Music Journal*, 1997. URL <http://www.jstor.org/stable/3681012>.
- [14] P.R. Cook. Din of an “iquity”: Analysis and synthesis of environmental sounds. In *Proceedings of the International Conference on Auditory Display (ICAD2007)*, pages 167–172, 2007.
- [15] Richard Corbett, Kees van den Doel, John E. Lloyd, and Wolfgang Heidrich. Timbrefields: 3d interactive sound models for real-time audio. *Presence: Teleoperators and Virtual Environments*, 16(6):643–654, 2007. doi: 10.1162/pres.16.6.643. URL <http://www.mitpressjournals.org/doi/abs/10.1162/pres.16.6.643>.
- [16] M. Desainte-Catherine and P. Hanna. Statistical approach for sound modeling. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, 2000.
- [17] A. Di Scipio. Synthesis of environmental sound textures by iterated nonlinear functions. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, 1999.
- [18] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics. *ACM Trans. Graph.*, 22:732–740, July 2003. ISSN 0730-0301. doi: <http://doi.acm.org/10.1145/882262.882339>. URL <http://doi.acm.org/10.1145/882262.882339>.
- [19] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Synthesizing sound from turbulent field using sound textures for interactive fluid simulation. In *Proc. of Eurographics*, pages 539–546, 2004.
- [20] Kees van den Doel. Physically based models for liquid sounds. *ACM Trans. Appl. Percept.*, 2:534–546, October 2005. ISSN 1544-3558. doi: <http://doi.acm.org/10.1145/1101530.1101554>. URL <http://doi.acm.org/10.1145/1101530.1101554>.
- [21] S. Dubnov and N. Tishby. Analysis of sound textures in musical and machine sounds by means of higher order statistical features. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*

- (ICASSP), volume 5, pages 3845–3848. IEEE, 1997. ISBN 0818679190. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=604726.
- [22] Shlomo Dubnov, Ziz Bar-Joseph, Ran El-Yaniv, Danny Lischinski, and Michael Werman. Synthesis of audio sound textures by learning and resampling of wavelet trees. *IEEE Computer Graphics and Applications*, 22(4):38–48, 2002.
 - [23] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision*, volume 2, page 1033, 1999.
 - [24] Aaron Einbond, Diemo Schwarz, and Jean Bresson. Corpus-based transcription as an approach to the compositional control of timbre. In *Proceedings of the International Computer Music Conference (ICMC)*, Montreal, QC, Canada, 2009.
 - [25] R. El-Yaniv, S. Fine, and N. Tishby. Agnostic classification of Markovian sequences. In *Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 465–471. MIT Press, 1998. ISBN 0262100762.
 - [26] Z.B.J.R. El-Yaniv, D.L.M. Werman, and S. Dubnov. Granular Synthesis of Sound Textures using Statistical Learning. In *Proceedings of the International Computer Music Conference (ICMC)*, 1999.
 - [27] D.P.W. Ellis, X. Zeng, and J.H. McDermott. Classifying soundtracks with audio texture features. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010. URL <http://www.ee.columbia.edu/~dpwe/pubs/EllisZM11-texture.pdf>.
 - [28] G. Essl and S. O’Modhrain. Scrubber: an interface for friction-induced sounds. In *Proceedings of the Conference for New Interfaces for Musical Expression (NIME)*, page 75. National University of Singapore, 2005.
 - [29] G. Fan and X.G. Xia. Wavelet-based texture analysis and synthesis using hidden Markov models. *IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Applications*, 50(1), 2003.
 - [30] Andy Farnell. *Designing Sound*. MIT Press, October 2010. ISBN 9780262014410. URL <http://mitpress.mit.edu/catalog/item/default.asp?type=2&tid=12282>.
 - [31] J.J. Filatriau and D. Arfib. Instrumental gestures and sonic textures. In *Proceedings of the International Conference on Sound and Music Computing (SMC)*, 2005.
 - [32] J.J. Filatriau, D. Arfib, and JM Couturier. Using visual textures for sonic textures production and control. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, 2006.
 - [33] N. Finney. Autonomous generation of soundscapes using unstructured sound databases. Master’s thesis, MTG, IUA–UPF, Barcelona, Spain, 2009. URL static/media/Finney-Nathan-Master-Thesis-2009.pdf.
 - [34] N. Finney and J. Janer. Soundscape Generation for Virtual Environments using Community-Provided Audio Databases. In *W3C Workshop: Augmented Reality on the Web*, June 15 - 16, 2010 Barcelona, 2010. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.168.4252&rep=rep1&type=pdfhttp://www.w3.org/2010/06/w3carl/>.
 - [35] M. Fröjd and A. Horner. Sound texture synthesis using an overlap-add/granular synthesis approach. *Journal of the Audio Engineering Society*, 57(1/2):29–37, 2009. URL <http://www.aes.org/e-lib/browse.cfm?elib=14805>.
 - [36] WW Gaver. How do we hear in the world? Explorations in ecological acoustics. *Ecological psychology*, 1993.
 - [37] T. Grill. Re-texturing the sonic environment. In *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound*, pages 1–7. ACM, 2010. URL <http://portal.acm.org/citation.cfm?id=1859805>.
 - [38] S. Guidati and Head Acoustics GmbH. Auralisation and psychoacoustic evaluation of traffic noise scenarios. *Journal of the Acoustical Society of America*, 123(5):3027, 2008.
 - [39] P. Hanna. *Statistical modelling of noisy sounds : spectral density, analysis, musical transformations and synthesis*. PhD thesis, Laboratoire de Recherche en Informatique de Bordeaux (LaBRI), 2003. URL <http://dept-info.labri.fr/~hanna/phd.html>.
 - [40] P. Hanna, N. Louis, M. Desainte-Catherine, and J. Benois-Pineau. Audio features for noisy sound segmentation. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR’04)*, pages 120–123, 2004.
 - [41] Pierre Hanna and Myriam Desainte-Catherine. A Statistical and Spectral Model for Representing Noisy Sounds with Short-Time Sinusoids. *EURASIP Journal on Advances in Signal Processing*, 2005(12):1794–1806, 2005. ISSN 1687-6172. doi: 10.1155/ASP.2005.1794. URL <http://www.hindawi.com/journals/asp/2005/182056.abs.html>.
 - [42] R. Hoskinson. *Manipulation and Resynthesis of Environmental Sounds with Natural Wavelet Grains*. PhD thesis, The University of British Columbia, 2002. URL https://www.cs.ubc.ca/grads/resources/thesis/May02/Reynald_Hoskinson.pdf.
 - [43] Reynald Hoskinson and Dinesh Pai. Manipulation and resynthesis with natural grains. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 338–341, Havana, Cuba, September 2001.
 - [44] I. Kauppinen and K. Roth. An Adaptive Technique for Modeling Audio Signals. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, 2001.
 - [45] Stefan Kersten and Hendrik Purwins. Sound texture synthesis with hidden markov tree models in the wavelet domain. In *Proceedings of the International Conference on Sound and Music Computing (SMC)*, Barcelona, Spain, July 2010.
 - [46] Anil Kokaram and Deirdre O’Regan. Wavelet based high resolution sound texture synthesis. In *Proceedings of the Audio Engineering Society Conference*, 6 2007. URL <http://www.aes.org/e-lib/browse.cfm?elib=13952>.
 - [47] M. Lagrange, B.L. Giordano, P. Depalle, and S. McAdams. Objective quality measurement of the excitation of impact sounds in a source/filter model. *Acoustical Society of America Journal*, 123:3746, 2008.
 - [48] Jung Suk Lee, Philippe Depalle, and Gary Scavone. Analysis / synthesis of rolling sounds using a source filter approach. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, Graz, Austria, September 2010.

- [49] L. Lu, S. Li, L. Wen-yin, H.J. Zhang, and Y. Mao. Audio textures. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, 2002. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.4586&rep=rep1&type=pdf>.
- [50] L. Lu, L. Wenyin, and H.J. Zhang. Audio textures: Theory and applications. *IEEE Transactions on Speech and Audio Processing*, 12(2):156–167, 2004. ISSN 1063-6676. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1284343.
- [51] A. Mallock. Sounds produced by drops falling on water. *Proc. R. Soc.*, 95:138–143, 1919.
- [52] Aymeric Masurelle. Gestural control of environmental texture synthesis. Rapport de stage de DEA ATIAM, Ircam–Centre Pompidou, Université Paris VI, 2011.
- [53] J.H. McDermott, A.J. Oxenham, and E.P. Simoncelli. Sound texture synthesis via filter statistics. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, October 18–21 2009.
- [54] Dylan Menzies. Phya and vfoley, physically motivated audio for virtual environments. *Proceedings of the Audio Engineering Society Conference*, 2010. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.9848&rep=rep1&type=pdf>.
- [55] Dylan Menzies. Physically Motivated Environmental Sound Synthesis for Virtual Worlds. *EURASIP Journal on Audio, Speech, and Music Processing*, 2011. URL <http://www.hindawi.com/journals/asmp/2010/137878.html>.
- [56] Nadine E. Miner and Thomas P. Caudell. Using wavelets to synthesize stochastic-based sounds for immersive virtual environments. *ACM Trans. Appl. Percept.*, 2:521–528, October 2005. ISSN 1544-3558. doi: <http://doi.acm.org/10.1145/1101530.1101552>. URL <http://doi.acm.org/10.1145/1101530.1101552>.
- [57] Nicolas Misdariis, Antoine Minard, Patrick Susini, Guillaume Lemaitre, Stephen McAdams, and Etienne Parizet. Environmental sound perception: Metadescription and modeling based on independent primary studies. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010. URL <http://articles.ircam.fr/textes/Misdariis10b/>.
- [58] A. Misra and P.R. Cook. Toward synthesized environments: A survey of analysis and synthesis methods for sound designers and composers. In *Proceedings of the International Computer Music Conference (ICMC)*, 2009.
- [59] A. Misra, P.R. Cook, and G. Wang. A new paradigm for sound design. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, 2006.
- [60] A. Misra, P.R. Cook, and G. Wang. TapeSTrea: Sound scene modeling by example. In *ACM SIGGRAPH 2006 Sketches*, page 177. ACM, 2006. ISBN 1595933646.
- [61] A. Misra, G. Wang, and P. Cook. Musical Tapestry: Recomposing Natural Sounds†. *Journal of New Music Research*, 36(4):241–250, 2007. ISSN 0929-8215.
- [62] William Moss, Hengchin (Yero) Yeh, Jeong-Mo Hong, Ming C. Lin, and Dinesh Manocha. Sounding Liquids: Automatic Sound Synthesis from Fluid Simulation. *ACM Transactions on Graphics*, 28(4):1–12, August 2009. ISSN 07300301. doi: [10.1145/1559755.1559763](http://doi.acm.org/10.1145/1559755.1559763). URL <http://portal.acm.org/citation.cfm?doid=1559755.1559763>.
- [63] E. Murphy, M. Lagrange, G. Scavone, P. Depalle, and C. Guastavino. Perceptual Evaluation of a Real-time Synthesis Technique for Rolling Sounds. In *Conference on Enactive Interfaces*, Pisa, Italy, 2008.
- [64] J.F. O’Brien, C. Shen, and C.M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 175–181. ACM New York, NY, USA, 2002.
- [65] S. O’Modhrain and G. Essl. PebbleBox and CrumbleBag: tactile interfaces for granular synthesis. In *Proceedings of the Conference for New Interfaces for Musical Expression (NIME)*, page 79. National University of Singapore, 2004.
- [66] D. O’Regan and A. Kokaram. Multi-resolution sound texture synthesis using the dual-tree complex wavelet transform. In *Proc. 2007 European Signal Processing Conference (EUSIPCO)*, 2007. URL <http://www.eurasip.org/Proceedings/Eusipco/Eusipco2007/Papers/A3L-B03.pdf>.
- [67] J.R. Parker and B. Behm. Creating audio textures by example: tiling and stitching. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 4:iv–317–iv–320 vol.4, May 2004. ISSN 1520-6149. doi: [10.1109/ICASSP.2004.1326827](http://doi.acm.org/10.1109/ICASSP.2004.1326827).
- [68] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the Cuidado project. Technical Report version 1.0, Ircam – Centre Pompidou, Paris, France, April 2004. URL http://www.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf.
- [69] Leevi Peltola, Cumhur Erkut, P.R. Cook, and Vesa Valimäki. Synthesis of hand clapping sounds. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1021–1029, March 2007. ISSN 1558-7916. doi: [10.1109/TASL.2006.885924](http://doi.acm.org/10.1109/TASL.2006.885924). URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4100694.
- [70] Gabriel Peyré. Oriented patterns synthesis. Technical report, Unité Mixte de Recherche du C.N.R.S. No. 7534, 2007. URL http://www.ceremade.dauphine.fr/preprints/consult/cmd_by_years.php.
- [71] Cecile Picard, Nicolas Tsingos, and François Faure. Retargeting Example Sounds to Interactive Physics-Driven Animations. In *AES 35th International Conference, Audio in Games*, London, UK, 2009.
- [72] K. Popat and R. W. Picard. Cluster-based probability model and its application to image and texture processing. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 6(2):268–84, January 1997. ISSN 1057-7149. doi: [10.1109/83.551697](http://doi.acm.org/10.1109/83.551697). URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=551697.
- [73] Xavier Rodet and Phillipe Depalle. A new additive synthesis method using inverse Fourier transform and spectral envelopes. In *Proceedings of the International Computer Music Conference (ICMC)*, October 1992.

- [74] Nicolas Saint-Arnaud. *Classification of Sound Textures*. PhD thesis, Université Laval, Quebec, MIT, 1995.
- [75] Nicolas Saint-Arnaud and Kris Popat. Analysis and synthesis of sound textures. In *in Readings in Computational Auditory Scene Analysis*, pages 125–131, 1995.
- [76] R. Murray Schafer. *The Soundscape*. Destiny Books, 1993. ISBN 0892814551. URL <http://www.amazon.com/Soundscape-R-Murray-Schafer/dp/0892814551>.
- [77] Diemo Schwarz. *Data-Driven Concatenative Sound Synthesis*. Thèse de doctorat, Université Paris 6 – Pierre et Marie Curie, Paris, 2004. URL <http://mediatheque.ircam.fr/articles/textes/Schwarz04a>.
- [78] Diemo Schwarz. Concatenative sound synthesis: The early years. *Journal of New Music Research*, 35(1):3–22, March 2006. Special Issue on Audio Mosaicing.
- [79] Diemo Schwarz. Corpus-based concatenative synthesis. *IEEE Signal Processing Magazine*, 24(2):92–104, March 2007. Special Section: Signal Processing for Sound Synthesis.
- [80] Diemo Schwarz and Norbert Schnell. Descriptor-based sound texture sampling. In *Proceedings of the International Conference on Sound and Music Computing (SMC)*, pages 510–515, Barcelona, Spain, July 2010.
- [81] Diemo Schwarz and Norbert Schnell. A modular sound descriptor analysis framework for relaxed-real-time applications. In *Proc. ICMC*, New York, NY, 2010.
- [82] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, and Sam Britton. Real-Time Corpus-Based Concatenative Synthesis with CataRT. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, pages 279–282, Montreal, Canada, September 2006.
- [83] Diemo Schwarz, Roland Cahen, and Sam Britton. Principles and applications of interactive corpus-based concatenative synthesis. In *Journées d'Informatique Musicale (JIM)*, GMEA, Albi, France, March 2008. URL <http://mediatheque.ircam.fr/articles/textes/Schwarz08a/index.pdf>.
- [84] G. Strobl. Parametric Sound Texture Generator. Msc thesis, Universität für Musik und darstellende Kunst, Graz; Technische Universität Graz, 2007. URL <http://en.scientificcommons.org/43580321>.
- [85] G. Strobl, G. Eckel, and D. Rocchesso. Sound texture modeling: A survey. In *Proceedings of the International Conference on Sound and Music Computing (SMC)*, 2006.
- [86] Patrick Susini, Stephen McAdams, Suzanne Winsberg, Yvan Perry, Sandrine Vieillard, and Xavier Rodet. Characterizing the sound quality of air-conditioning noise. *Applied Acoustics*, 65-8:763–790, 2004. URL <http://articles.ircam.fr/textes/Susini04b/>.
- [87] N. Tsingos, E. Gallo, and G. Drettakis. Perceptual audio rendering of complex virtual environments. In *ACM SIGGRAPH 2004 Papers*, pages 249–258. ACM, 2004.
- [88] A. Valle, V. Lombardo, and M. Schirosa. Simulating the Soundscape through an Analysis/Resynthesis Methodology. *Auditory Display*, pages 330–357, 2010.
- [89] Andrea Valle, Vincenzo Lombardo, and Mattia Schirosa. A graph-based system for the dynamic generation of soundscapes. In *Proceedings of the 15th International Conference on Auditory Display*, pages 217–224, Copenhagen, 18–21 May 2009.
- [90] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01*, pages 537–544, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X. doi: <http://doi.acm.org/10.1145/383259.383322>. URL <http://doi.acm.org/10.1145/383259.383322>.
- [91] C. Verron, M. Aramaki, R. Kronland-Martinet, and G. Pallone. *Spatialized Synthesis of Noisy Environmental Sounds*, pages 392–407. Springer-Verlag, 2010. URL <http://www.springerlink.com/index/J3T5177W11376R84.pdf>.
- [92] C. Verron, M. Aramaki, R. Kronland-Martinet, and G. Pallone. Contrôle intuitif d'un synthétiseur d'environnements sonores spatialisés. In *10eme Congres Français d'Acoustique*, 2010. URL <http://cfa.sfa.asso.fr/cd1/data/articles/000404.pdf>.
- [93] Charles Verron. *Synthèse immersive de sons d'environnement*. PhD thesis, Université Aix-Marseille I, 2010.
- [94] X. Zhu and L. Wyse. Sound texture modeling and time-frequency LPC. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, volume 4, 2004.

VECTOR PHASESHAPING SYNTHESIS

Jari Kleimola*, Victor Lazzarini†, Joseph Timoney†, Vesa Välimäki*

*Aalto University School of Electrical Engineering, Espoo, Finland

†National University of Ireland, Maynooth, Ireland

jari.kleimola@aalto.fi, victor.lazzarini@nuim.ie, joseph.timoney@nuim.ie
vesa.valimaki@tktk.fi

ABSTRACT

This paper introduces the Vector Phaseshaping (VPS) synthesis technique, which extends the classic Phase Distortion method by providing flexible means to distort the phase of a sinusoidal oscillator. This is achieved by describing the phase distortion function using one or more breakpoint vectors, which are then manipulated in two dimensions to produce waveshape modulation at control and audio rates. The synthesis parameters and their effects are explained, and the spectral description of the method is derived. Certain synthesis parameter combinations result in audible aliasing, which can be reduced with a novel aliasing suppression algorithm described in the paper. The extension is capable of producing a variety of interesting harmonic and inharmonic spectra, including for instance, formant peaks, while the two-dimensional form of the control parameters is expressive and is well suited for interactive applications.

1. INTRODUCTION

Abstract sound synthesis techniques have had a long history of development. Since the introduction of digital waveshaping in Risset's catalogue of computer instruments [1] in the late 1960s, subsequent theories related to non-linear distortion synthesis methods, such as FM, Discrete Summation Formulae (DSF) and others [2] [3] [4] [5] [6] [7] have emerged. Recently, the area has been revitalised with work on adaptive techniques [8] [9] [10], as well as new non-linear [11] and audio feedback methods [12] [13].

In this paper, we will start from an established non-linear Phase Distortion (PD) synthesis method [14], and propose three extensions to it in order to distort a sinusoidal waveform in a more complex manner: the inflection point is described as a two-dimensional vector, the phase distortion function is defined with multiple inflection points, and the modulation rate of the inflection points is raised to audio frequencies. These extensions allow a wider sonic palette to be extracted from the method, as well as more flexible control over the spectral changes. The new technique is named *Vector Phaseshaping* (VPS) synthesis.

Phaseshaping [15] [16] can be understood as a generalisation of the idea of PD, which in turn can be seen as a type of complex-wave phase modulation [11]. Phaseshaping is also related to non-linear waveshaping [17], but has some advantages over it. One of them is that in phaseshaping the use of non-smooth shaping functions does not necessarily imply the presence of audible aliasing, which is more or less inevitable in waveshaping. In addition, it is possible to mitigate the effects of aliasing, as will be explored later in this paper. Finally, waveshaping – as it is based on a non-linear amplification effect – requires care in terms of gain scaling to be usable. This is not, in general, a requirement for phaseshaping.

After a brief introduction to the original PD synthesis technique in Section 2, this paper is organized as follows. Section 3 introduces the VPS method, defines its multi-point vectorial extension, derives its spectral description, and proposes a novel alias-suppression method. Section 4 explores control rate modulation of the vector in 1-D, 2-D, and multi-vector configurations, while Section 5 complements this at audio rates. Finally, Section 6 concludes.

2. PHASE DISTORTION SYNTHESIS

The classic PD synthesis technique is defined by equations

$$s(n) = -\cos\{2\pi\phi_{pd}[\phi(n)]\}, \quad (1)$$

$$\phi_{pd}(x) = \begin{cases} \frac{1}{2}\frac{x}{d}, & 0 \leq x \leq d \\ \frac{1}{2}\left[1 + \frac{(x-d)}{(1-d)}\right], & d < x < 1, \end{cases} \quad (2)$$

where n is the sample number and d is the point of inflection (see Fig. 1). In this case, equation (2) is a phaseshaper acting on an input signal $\phi(n)$. This is a trivial sawtooth wave with frequency f_0 and sampling rate f_s , and given by

$$\phi(n) = \left[\frac{f_0}{f_s} + \phi(n-1)\right] \bmod 1, \quad (3)$$

which is same as the phase signal used in a standard table lookup oscillator, for instance. The mod 1 operator can be defined as

$$x \bmod 1 \triangleq x - \lfloor x \rfloor, x \in \mathbb{R}. \quad (4)$$

In this particular phaseshaper, the point of inflection d determines the brightness, the number of harmonics, and therefore the shape of the output signal. The closer d is to 0 or to 1, the brighter the signal (and more prone to audible aliasing). At $d = 0.5$, there is no change in the phase signal as the shaper function is linear. In fact, there is a symmetry condition around $d = 0.5$, with the output being based on a falling shape with $0 \leq d < 0.5$ and a rising shape with $0.5 < d \leq 1$. By varying d , we can get an effect that is similar to changing the cutoff frequency of a low-pass filter. The PD equation can also be cast as a case of complex-wave phase modulation, as discussed in [11] and [15].

3. VECTOR PHASESHAPING SYNTHESIS

This section introduces the VPS method, which is a new extension to the phase distortion synthesis technique described above. In classic PD, the inflection point d controls the x-axis position of the phase distortion function bending point, which traces the thin

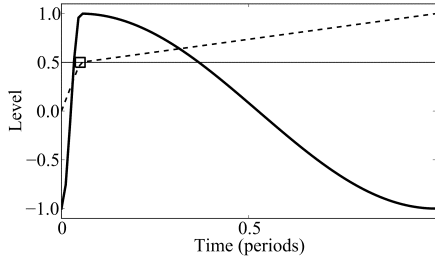


Figure 1: Classic PD sawtooth waveform (thick), phaseshaping function (dashed), and inflection point path (thin horizontal line). The inflection point $d = 0.05$ is marked with a square.

horizontal path shown in Fig. 1. Thus, the vertical position of the bend is fixed at 0.5. VPS synthesis releases this constraint and expresses the inflection point as a two-dimensional vector

$$p = (d, v), \quad (5)$$

where d is the horizontal, $0 \leq d \leq 1$, and v is the vertical position of the bending point. The two-dimensional phase distortion function is given by

$$\phi_{\text{VPS}}(x) = \begin{cases} \frac{vx}{d}, & 0 \leq x \leq d \\ (1-v)\frac{(x-d)}{(1-d)} + v, & d < x < 1, \end{cases} \quad (6)$$

which reduces to the classic form of equation (2) when $v = 0.5$.

To gain an understanding of the effect of v , consider first setting $p = (0.5, 0.5)$, which, when applied to the waveshaper of equation (1), produces an undistorted inverted cosine waveform as shown in Fig. 2(a). As v is then raised towards unity, the slope of the first segment of equation (6) is increased as well, and the waveshaped output of the first segment grows towards a full-cycle sinusoid (Fig. 2(b), before period 0.5). Consequently, the slope of the second segment becomes less steep, and at $v = 1$, it produces the static portion of the waveform depicted in Fig. 2(c).

The bandwidth of the spectrum grows from a single harmonic to the form shown in Fig. 2(b), and then shrinks towards that of Fig. 2(c). Since the latter waveform resembles a half-wave rectified sinusoid, its spectrum consists of odd harmonics, with a strong additional second harmonic.

On the other hand, if v is decreased from 0.5 towards 0, the slope of the first segment decreases while that of the second one increases. The waveforms of Fig. 2 become then reversed in time, and therefore, identical spectra are obtained for v values that are symmetric around 0.5. Since both vertical and horizontal domains of the inflection vector are symmetric around this position, the point $p_0 = (0.5, 0.5)$ defines the center location of the two-dimensional VPS parameter space. However, when either d or v is offset from its center position, this symmetry property (of the other parameter) is no longer sustained. This leads to some interesting characteristics that will be discussed next.

When $v = 1$, as in Fig. 2(c), d controls the duty width of the produced waveform. The pulse width increases with d , from a narrow impulse up to a full-cycle sinusoid at $d = 1$. Pulses of various widths can then be constructed with $p = (d, 1)$ and $0 < d \leq 0.5$, or $p = (d, 0)$ and $0.5 \leq d < 1$ (these are symmetrically-related vectors), as seen in Fig. 3. Transitions between various pulse widths and other waveshapes can be smoothly created by interpolating the vector values. These effects will be

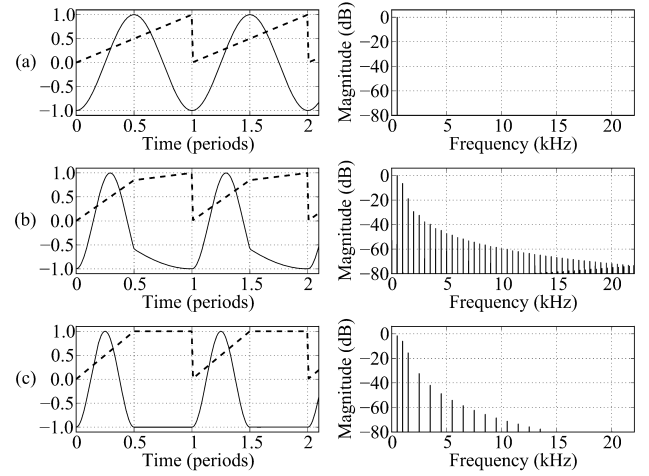


Figure 2: VPS waveforms and spectra of (a) $p=(0.5, 0.5)$, (b) $p=(0.5, 0.85)$, and (c) $p=(0.5, 1)$. $f_0 = 500$ Hz and $f_s = 44100$ Hz, as in all examples of this paper.

particularly interesting when vectors are subjected to modulation. Classic oscillator effects such as pulse-width modulation will be easily implemented by modulating the d value of the vector with a low-frequency oscillator (LFO).

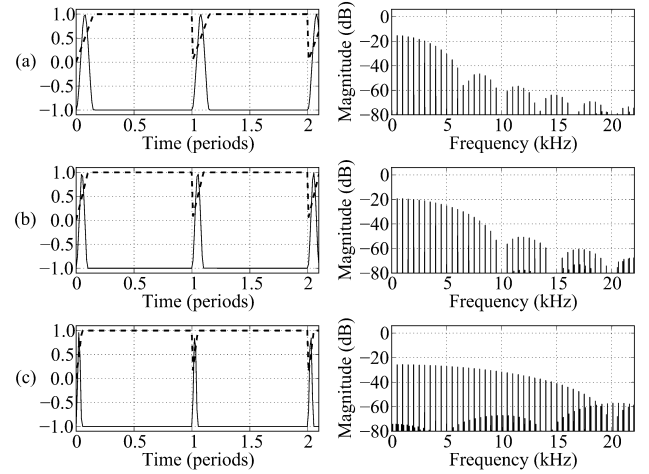


Figure 3: VPS pulse waveforms and spectra of (a) $p=(0.15, 1)$, (b) $p=(0.1, 1)$, and (c) $p=(0.05, 1)$.

Certain combinations of v and d will always produce sinusoids. For instance, $v = 1, 2, 3 \dots n$ and $d = 1$ forms a single-segment linear phaseshaper, and similarly, $v = d = 0.5$ (i.e., the trivial form) is linear, but there are other cases. For instance, sinusoids will be produced with $v = 1.5$ and $d = 0.75$, and with $v = 3$ and $d = 0.6$. The general form of this, for $0 < d < 1$, is

$$\left| \frac{v}{d} \right| = \left| \frac{1-v}{1-d} \right|, v/d \in \mathbb{Z}. \quad (7)$$

This is because the derivative of the phase on both sides of the inflection point has the same absolute value (it might only differ in sign). Due to the use of a cosine wave, which is an even function,

the change of sign of the derivative at the inflection point happens to be of no consequence. The pitch of the produced sinusoid will be equivalent to $|v/d|f_0$, the fundamental frequency times the absolute value of the phase derivative. The presence of these sinusoid cases means that, when changing the waveshapes by manipulating the vector, there will be loci of p where all components suddenly vanish, leaving a single harmonic sounding. This effect can be quite dramatic and of musical interest.

3.1. Synthesising Formants

At $d = 0.5$ and with $v \geq 1.5$ VPS produces formants, which are quite prominent when centred around exact harmonics of the fundamental. In such situations, the spectrum consists of five harmonics around a central frequency and of sidebands of odd order harmonics, as shown in Fig. 4(a). As can be seen, the magnitude of the emphasised frequency region is strong in comparison to the sidebands, which is a useful property in vocal and resonant filter synthesis applications.

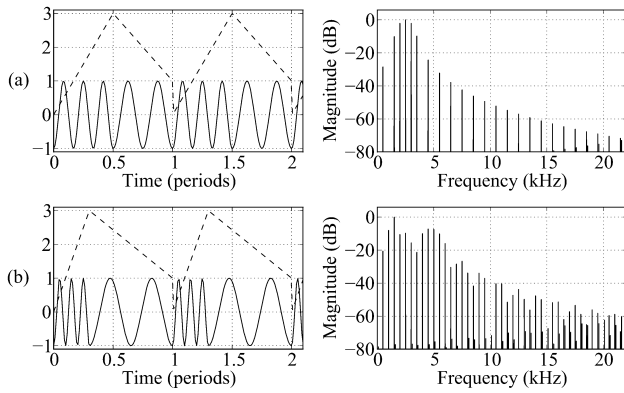


Figure 4: (a) VPS formant, $p = (0.5, 3)$, and (b) multiple formants ($d \neq 0.5$).

In these cases, the ratio of the formant centre frequency f_f and the fundamental frequency f_0 is defined as

$$\frac{f_f}{f_0} = 2v - 1, \quad (8)$$

so formants will be centred on exact harmonics of f_0 when $v = 1.5, 2, 2.5, \dots$

Offsetting d from its central position spreads the formant across the digital baseband as shown in Fig. 4(b). As in classic PD, d controls the spectral brightness of the timbre, and as it approaches 0 or 1, there will be an increased amount of aliasing in the output spectra. This is due to the high-frequency periods in the shorter segment. Aliasing can also happen when equation (8) does not yield integral values. The reason now is that there will be incomplete periods, i.e., discontinuities in the waveform itself or in its first derivative.

One possible way to counteract this last effect has been demonstrated in other techniques, such as PAF [7] and phase-synchronous ModFM [18]. These techniques also share a similar problem whereby if the formant frequency is not an exact multiple of the fundamental, discontinuities in the output waveform can occur. The solution is to use two oscillators, whose formant frequencies are tuned to adjacent multiples of the fundamental around the exact formant

frequency that is required. A simple linear crossfading of the two output signals will generate a peak at the target formant centre. This allows us to sweep the spectrum smoothly with no aliasing noise due to this particular effect. To achieve this, we define an interpolation gain a that is dependent on the fractional part of $f_f : f_0$,

$$a = [2v - 1] \bmod 1, \quad (9)$$

and then use it to scale the two VPS signals, $s_1(n)$ and $s_2(n)$, employing inflection vectors $p_1 = (0.5, v)$ and $p_2 = (0.5, v + 0.5)$, respectively, with $v > 1$ and $2v - 1 \in \mathbb{Z}$ to obtain the output signal $y(n)$:

$$y(n) = (1 - a)s_1(n) + as_2(n). \quad (10)$$

3.2. Aliasing Suppression

The aliasing produced by the incomplete periods may also be suppressed by exploiting a novel single oscillator algorithm, which modifies the phasemapper when $\phi_{vps}[\phi(n)] > \text{floor}(v)$, i.e., when the phase is inside the incomplete period. The modified phasemapper is given by

$$\phi_a(x) = \begin{cases} \frac{p \bmod 1}{2b}, & 0 < b \leq 0.5 \\ \frac{p \bmod 1}{b}, & 0.5 < b < 1, \end{cases} \quad (11)$$

where $p = \phi_{vps}(x)$ and $b = v \bmod 1$. When applied to equation (1), the incomplete segment is rendered as a smooth full-cycle sinusoid, which is then scaled and offset in relation to $c = \cos(2\pi b)$:

$$s_a(n) = \begin{cases} [(1 - c)s(n) - 1 - c]/2, & 0 < b \leq 0.5 \\ [(1 + c)s(n) + 1 - c]/2, & b > 0.5, \phi_a(x) > 0.5, \end{cases} \quad (12)$$

Fig. 5 shows that the aliasing present in the trivial VPS form (Fig. 5(a)) is reduced substantially when processed with this algorithm (Fig. 5(b)). This is achieved at the cost of reduced high-end spectral content.

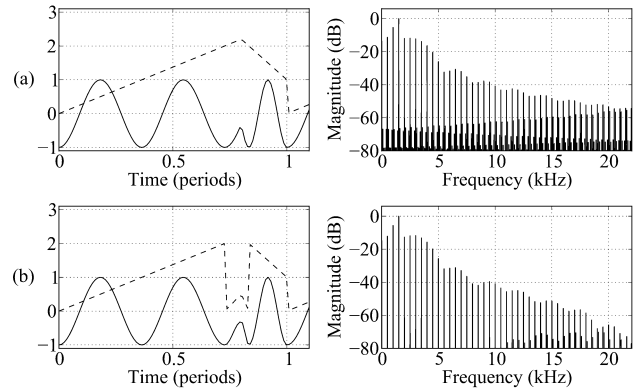


Figure 5: (a) Trivial VPS timbre and (b) its alias-suppressed form. $p = (0.8, 2.2)$.

3.3. Notes on the Derivation of the VPS Spectrum

Given that VPS, as an extension of PD, is effectively a complex form of Phase Modulation (PM) synthesis, it is reasonable to expect that we would be able to derive a closed-form expression for

its spectrum. In order to do this, we would put the technique in terms of a carrier phase, to which a given modulation function is applied. In [11] this is implemented for the PD modulation function of equation (2) using a complex modulating wave PM spectrum derivation [19]. This yields an expression with many terms based on a product series of Bessel coefficients that is very unwieldy and difficult to handle.

For simple geometric PM functions (such as the ones found in VPS), it is, however, possible to obtain an alternative spectral description that avoids the use of Bessel coefficients. This derivation method is similar to the one in [20]. For a VPS function with v and d , we can have a corresponding PM function $M(x) = \pi d\{2[\phi_{\text{VPS}}(x + dx) - x - dx] - 1\}$, $-\pi \leq x \leq \pi$, such that,

$$-\cos\{2\pi\phi_{\text{VPS}}[\phi(t)]\} = -\cos\{\omega t + M(t) + \phi_d\} = \sin(\omega t + \phi_d)\sin[M(t)] - \cos(\omega t + \phi_d)\cos[M(t)], \quad (13)$$

$$\cos[M(t)] = \begin{cases} \cos(y\theta), & 0 \leq \theta \leq \pi x \\ \cos[\frac{xy(\pi-\theta)}{1-x}], & \pi x < \theta \leq \pi, \end{cases} \quad (14)$$

$$\cos[M(t)] = \cos[-M(t)],$$

and

$$\sin[M(t)] = \begin{cases} \sin(y\theta), & 0 \leq \theta \leq \pi x \\ \sin[\frac{xy(\pi-\theta)}{1-x}], & \pi x < \theta \leq \pi, \end{cases} \quad (15)$$

$$\sin[M(t)] = -\sin[-M(t)],$$

where t is time, $\phi_d = 2\pi d$, $x = d/2$, and $y = v(1 - d)/2$.

Because $\cos[M(t)]$ and $\sin[M(t)]$ are even and odd, respectively, in order to compute their Fourier series, we only need half periods, as defined by equations (14) and (15). Using these series and the right-hand side of equation (13), we can obtain a description of the VPS spectrum as

$$s(t) = -b'_0 \cos(\omega t + \phi_d) + \sum_{n=1}^{\infty} \frac{c_n - b_n}{2} \cos(\omega t[n - 1] + \phi_d) - \frac{c_n + b_n}{2} \cos(\omega t[1 + n] + \phi_d) \quad (16)$$

with

$$b_n = \frac{2}{\pi} \int_0^{\pi x} \cos(y\theta) \cos(n\theta) d\theta + \frac{2}{\pi} \int_{\pi x}^{\pi} \cos(\frac{xy(\pi-\theta)}{1-x}) \sin(n\theta) d\theta, \quad (17)$$

$$c_n = \frac{2}{\pi} \int_0^{\pi x} \sin(y\theta) \sin(n\theta) d\theta + \frac{2}{\pi} \int_{\pi x}^{\pi} \sin(\frac{xy(\pi-\theta)}{1-x}) \sin(n\theta) d\theta, \quad (18)$$

and $b'_0 = \frac{b_0}{2}$.

With these expressions, it is possible to derive the spectrum of single inflection point VPS. However, in our studies, we have found that the VPS method is at times more simply described in terms of waveform morphologies linked to the geometry of phase-shaping functions. Some of these are described in Table 1.

3.4. Multiple Inflection Points

Finally, it is interesting to consider the possibility of more than one inflection points. This can be used to obtain, for instance, square wave-like output signals. Consider for instance the use of three vectors $p_0 = (d_0, v_0)$, $p_1 = (d_1, v_1)$, and $p_2 = (d_2, v_2)$. With $p_0 = (0.1, 0.5)$, $p_1 = (0.5, 0.5)$, and $p_2 = (0.6, 1)$, we have a wave that approximates a square shape (see Fig. 6). These three points, however, can be freely manipulated to provide a variety of waveforms.

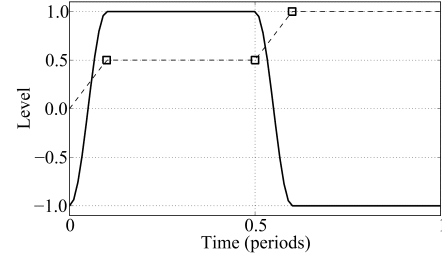


Figure 6: Square-like waveform (thick) produced by a phase-shaping function (dashed) with three inflection points at $p_0 = (0.1, 0.5)$, $p_1 = (0.5, 0.5)$, and $p_2 = (0.6, 1)$.

The general form for the phaseshaping function in Multiple Vector Phaseshaping with N inflection points p_0, p_1, \dots, p_{N-1} is:

$$\phi_{\text{mvps}}(x) = \begin{cases} \frac{v_0 x}{d_0}, & x < d_0 \\ (v_n - v_{n-1}) \frac{(x - d_{n-1})}{(d_n - d_{n-1})} + v_{n-1}, & d_{n-1} \leq x < d_n \\ \dots & \\ (1 - v_{N-1}) \frac{(x - d_{N-1})}{(1 - d_{N-1})} + v_{N-1}, & x \geq d_{N-1} \end{cases} \quad (19)$$

which reduces to equation (6) for $N = 1$.

4. ADAPTIVE VECTOR CONTROL

VPS synthesis provides rich spectra of various forms. In order to seize the musical potential from the method, we need to provide adaptive controls [21] to the vector parameters. In this section, we will examine various possibilities arising from this.

4.1. One-dimensional Control

The simplest means of adaptive control over waveform shapes is provided by varying one of the vector parameters, whilst holding the other constant. For instance, keeping $v = 0.5$ and varying d provides an emulation of a low-pass filter sweep, as the spectrum gets richer with d close to 0 or 1. Keeping $d = 0.5$ and varying v also provides a similar effect, but now with a resonant peak at $(2v - 1)f_0$, as discussed earlier. Other fixed values of d and v will create transitions between the various characteristics outlined in the previous section.

4.2. Two-dimensional Low-frequency Modulation

It is with two-dimensional adaptive control, however, that VPS synthesis becomes a very original proposition. This can be performed by a joystick or an x-y controller. Transitions between a

Table 1: Some VPS morphologies.

$p(d, v)$	Waveshape	Spectrum	Figure
$(0.5, 1)$	half sinusoid	missing some even harmonics	2(c)
$(0.5, < 1)$	distorted half sinusoid	steep spectral slope	2(b)
$(0.5, > 1)$	varying-period sinusoids	peaks at $(2v - 1)f_0$	4(a)
$(< 0.5, 1)$	pulse-like	spectrum gets richer with $d \rightarrow 0$	3
$(< 0.5, > 1)$	varying-period distorted sinusoids	multiple formant peaks	4(b)
$(< 0.5, 0.5)$	sawtooth-like	more gradual spectral slope	1

variety of waveshapes (e.g., with those summarised in Table 1) can be easily achieved, and this can provide a great source of timbral expression in musical performance.

This facility can be extended by the use of a 2-D LFO. This can take the form of two separate oscillators, controlling the two parameters d and v , i.e., the rectangular coordinates of the vector, or its polar representation, the angle and magnitude. The oscillators can exhibit different waveshapes and frequencies.

4.3. Lissajous Modulation

An interesting application of a 2-D LFO can be achieved by synchronising the phase of the two oscillators, so that the path of the modulated inflection point forms a *Lissajous figure* [22]. This is achieved by combining the two modulator signals in the following system of equations:

$$\begin{cases} d = A_d \{0.5 + 0.5 \cos(\omega_d + \theta)\} \\ v = A_v \{0.5 + 0.5 \cos(\omega_v)\}, \end{cases} \quad (20)$$

with $0 \leq A_d \leq 1$ and $A_v \geq 0$.

Various interesting 2-D modulation shapes can then be obtained. With $\omega_d = \omega_v$ and $\theta = \pi/2$ we can create circular or elliptical paths. If the two LFO frequencies are different, $\omega_d = n\omega_v$ or $\omega_v = m\omega_d$ and $\theta = \pi/2$, we will have n vertical or m horizontal ‘rings’ ($m, n \in \mathbb{Z}$ and $m > 1$). By varying θ , we can also collapse the path into a straight diagonal line ($\omega_d = \omega_v$ and $\theta = 0$, ascending; or $\theta = 0.5$, descending). Complex paths can be created by varying these parameters. Fig. 7 shows various combinations of these Lissajous modulation paths, while Fig. 8 shows a spectrogram of a Lissajous-modulated VPS timbre. In addition, a second-order modulator can be employed to control these parameters for a cyclical modulation path transformation.

4.4. Multi-vector Modulation

Finally, we must consider the possibilities of multi-vector modulation. Here, the multiplication of parameters might pose a problem for controller mapping. In addition, the horizontal component of each vector will work on a limited range, which will depend on the positions of neighboring inflection points. This is required so that the phaseshaping function remains single-valued. However, bounds for the vertical component work as before.

One solution to this issue is to use a single controller (such as a 2-D LFO or a joystick), which would determine the positions of all inflection points by a mapping matrix. The advantage of this is that principles developed for single vectors, such as Lissajous modulation, can be easily extended to this case.

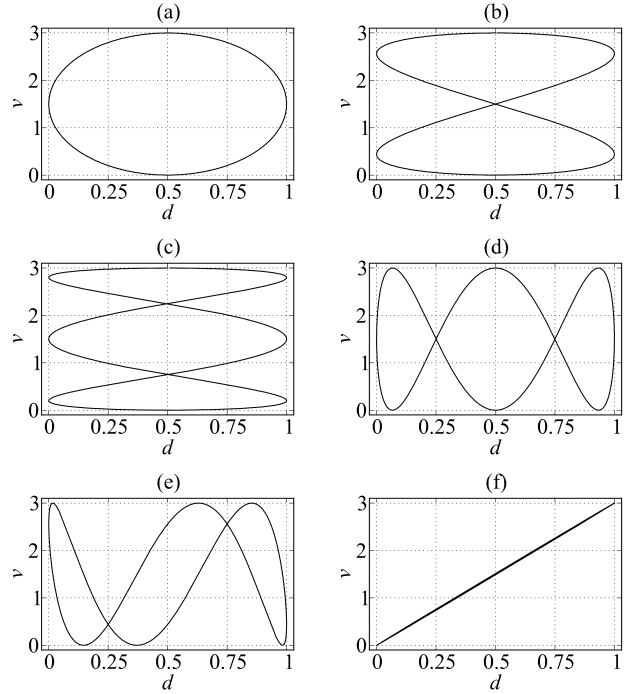


Figure 7: Lissajous modulation, with $A_d = 1$ and $A_v = 3$: (a) $\omega_d = \omega_v$ and $\theta = \pi/2$; (b) $\omega_d = 2\omega_v$ and $\theta = \pi/2$; (c) $\omega_d = 3\omega_v$ and $\theta = \pi/2$; (d) $\omega_v = 3\omega_d$ and $\theta = \pi/2$; (e) $\omega_v = 3\omega_d$ and $\theta = \pi/4$; (f) $\omega_d = \omega_v$ and $\theta = 0.01$.

To explore independent modulation of vectors, a solution can be found in multi-touch controllers, where each inflection point can be determined by finger position. Given that each segment of the phaseshaping function will be independent, this type of adaptive control can be used to create wave sequencing effects.

5. AUDIO-RATE WAVESHAPE MODULATION

Other types of complex spectra are obtained by extending the application of modulation to audio frequencies. As in the previous section, we will first look at the modulation of individual vector components, then study the combination of the two dimensions.

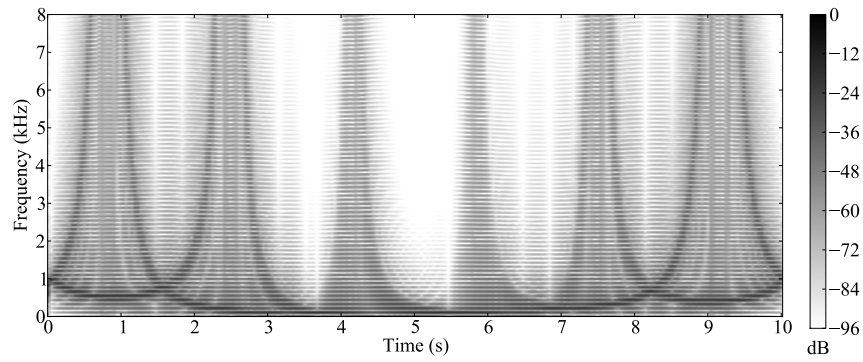


Figure 8: Spectrogram of VPS with Lissajous modulation, $\omega_d = 3\omega_v$, $\theta = \pi/2$, $A_d = 1$ and $A_v = 5$.

5.1. One-dimensional Vector Modulation

Starting with the original PD arrangement, with $v = 0.5$, and modulating d , we observe two basic types of output for $f_m = f_0$ (with f_m as the modulator frequency):

1. Single-sided modulation, where $0 \leq d \leq 0.5$ or $0.5 \leq d \leq 1$, using an inverted cosine modulator produces a spectrum that decays more abruptly than $1/f$ and a sawtooth-like waveshape, see Fig. 9(a). The bigger the phase difference between the modulator and an inverted cosine, means the brighter the spectrum. The spectrum is brightest overall with a cosine modulator.
2. Double-sided modulation, where $0 \leq d \leq 1$, we observe a peak at the second harmonic, and a more gradual decay in the spectral envelope, with a cosine or inverted cosine modulator. Discontinuities in the waveform reset also produce substantial aliasing, see Fig. 9 (b). The modulator phase also has an effect: high frequency components will be substantially attenuated and the peak at the second harmonic disappears when the modulator is a sine wave, see Fig. 9(c).

By reducing the modulation amount, less components will be generated, so this parameter can be used as a timbre control. In general, a fundamental difference between static or low-frequency modulated and audio-rate modulated VPS is that in the latter case, the phaseshaping functions are not based on linear segments, as seen in Fig. 9.

On the other hand, if we hold d static at 0.5 and vary v , sinusoidally, $f_m = f_0$ and $0 \leq v \leq k$, we will have a bright spectrum that depends on the modulation width k . Higher values of k will distribute the energy more evenly and produce a richer spectrum. At the highest values of k , spectral peaks will be less pronounced. Interestingly, the output will also be quasi-bandlimited, so it is possible to suppress aliasing by keeping k under control. Modulator phase will also affect the output waveform shape and spectrum. Fig. 10 shows three different cases of this type of modulation.

Finally, we must consider the cases where $f_m \neq f_0$. When $f_m = n f_0$, $n \in \mathbb{Z}$, the spectrum will be harmonic and generally increasing in brightness with n . If $f_m \ll f_0$, the perceived fundamental will not be equivalent to f_0 anymore. In cases where $f_m = f_0/n$, $n \in \mathbb{Z}$, we will have a harmonic spectrum with f_m as the fundamental. When f_m/f_0 is not a ratio of small numbers or is irrational, we will have an inharmonic spectrum. The reason for this is that the output will be composed of a fast sequence

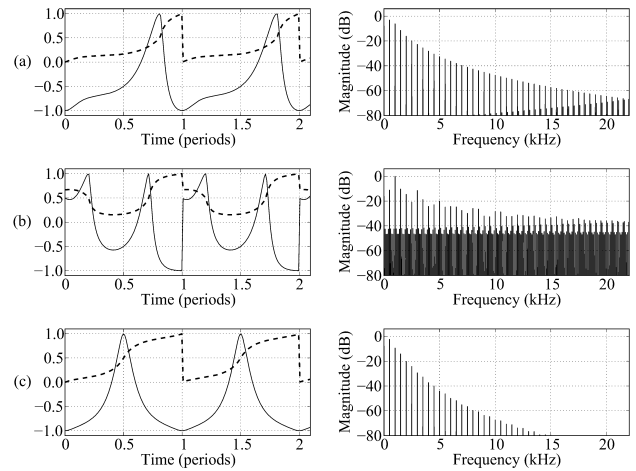


Figure 9: Audio-rate shape modulation of vector component d , $f_m = f_0$ (output, solid line; phase, dashed line): (a) Single-sided, inverted cosine modulator; (b) double-sided, inverted cosine modulator; and (c) double-sided, sine modulator.

of different waveshapes, which will not be fused into a periodic pattern. Interesting cases happen when f_m and f_0 are very close, but not exactly the same value. In these cases, the spectrum will be harmonic and we will perceive a cyclically changing timbral pattern whose period is $1/(f_m - f_0)$. The spectrogram of such a tone is shown in Fig. 11, where $f_m = f_c - 0.1$. These observations regarding the modulation frequency are similarly applied to the two-dimensional cases discussed below.

5.2. Two-dimensional Vector Modulation

Two-dimensional audio-rate modulation is more conveniently implemented using the Lissajous arrangement introduced in the previous section. In the case of audio-rate modulation, its parameters will be modulation frequency f_m , horizontal to vertical frequency ratio $\omega_d : \omega_v$ (which scales the modulation frequency for each component), horizontal phase difference θ , and horizontal and vertical modulation width, A_d and A_v , respectively. Fig. 12 shows three examples of different timbres produced by varying the Lissajous parameters, all of them with $f_m = f_c$, defining three dif-

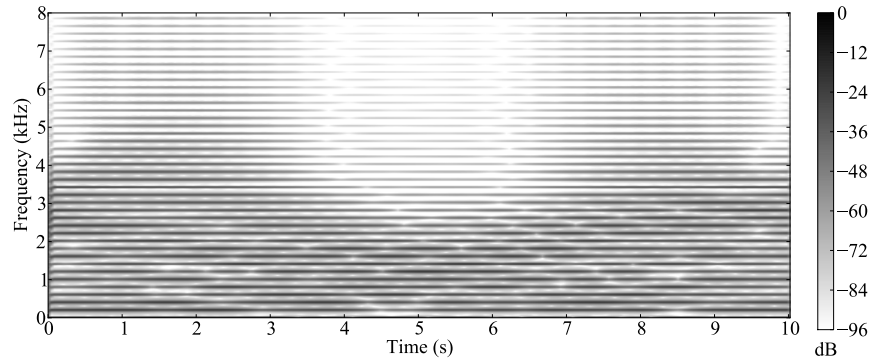


Figure 11: Spectrogram of an audio-rate modulated VPS timbre showing a cyclically changing spectrum, $d = 0.5$ with v modulated by an inverted cosine, $k = 5$, $f_c = 500$ Hz, and $f_m = f_c - 0.1$.

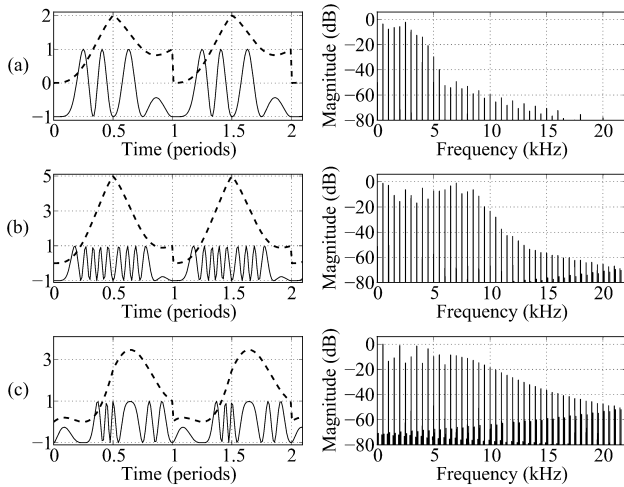


Figure 10: Audio-rate shape modulation of vector component v , $f_m = f_0$ (output, solid line; phase, dashed line): (a) modulator width $k = 2$, (b) $k = 5$ (cosine phase) and (c) $k = 2$ (sine phase).

ferent modulation paths: circular, three vertical rings and nearly linear.

The advantages to the use of Lissajous modulation is that it is possible to identify, in general lines, a particular modulation path with a given tone spectrum. Therefore, the principles of defining morphologies, which was done in Table 1 for the VPS tones, can be extended to the more complex 2-D audio-rate shape-modulated timbres. This could be the basis for the selection of desired timbral qualities, with a simpler and more compact parameter set than the case of independent modulators for the two dimensions. For instance, it is clear from Fig. 12 that some paths will be more prone to aliasing (e.g., Fig. 12 (c)), whereas others provide a cleaner spectrum.

There is no doubt that 2-D audio-rate modulation will inevitably produce aliasing of some kind. How objectionable this might be is probably a better question. From a musical point of view, what we observe in the output is that some types of modulation will generate aliasing that is perceptually a form of bright broadband noise, and that is a sonority that is in some cases desirable. In this case, a

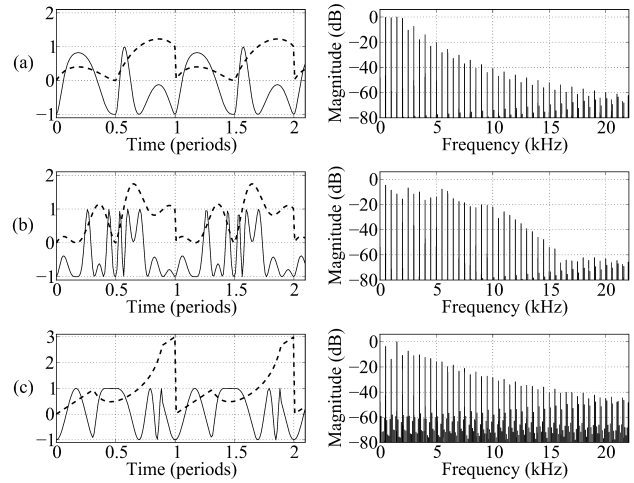


Figure 12: Lissajous audio-rate shape modulation of vector component v , $f_m = f_0$, $A_d = 1$ and $A_v = 3$ (output, solid line; phase, dashed line): (a) $\omega_d = \omega_v$ and $\theta = -\pi/2$ (circular path); (b) $\omega_v = 3\omega_d$ and $\theta = -\pi/2$ (three vertical rings); (c) $\omega_d = \omega_v$ and $\theta = 0.01$ (nearly linear path).

technique for a synthesiser with wide musical applications might embrace the production of some amount of aliasing as one of its characteristics, rather than consider it to be a defect.

5.3. Multi-vector Manipulation

Completing the types of audio-rate shape modulation that are possible with VPS, we have multi-vector manipulation. Here, at least four scenarios can be described:

- One-dimensional manipulation of a single vector (other vectors constant)
- One-dimensional manipulation of multiple vectors.
- Two-dimensional manipulation of a single vector.
- Two-dimensional manipulation of multiple vectors.

Regarding these different cases, a few general lines can be discerned. The first case of one-dimensional modulation is effectively

an extension of the single-vector case, and it is possible to apply the principles discussed above to it. Similarly, two-dimensional manipulation can be seen as an extension of ideas previously discussed in this paper, both in low-frequency modulation of multiple vectors and in audio-rate Lissajous modulation.

The most complex case to handle, however, is possibly one-dimensional manipulation of multiple vectors. Various sub-cases can arise from this, since each inflection point can be modulated in either direction. It is possible to simplify our approach and use a single modulator that is mapped to different vectors/directions, as in the case of the matrix mapping discussed earlier on in Section 3.4. The ranges of the horizontal component of each point will have to be scaled properly so that no overlap occurs. In any case, multi-vector manipulation is by far the most complex case of VPS synthesis and it requires a detailed study that is left as future work.

6. CONCLUSIONS

This paper introduced the technique of Vector Phaseshaping synthesis as an extension of the well-known Phase Distortion method. Its main characteristics and spectral description were defined in detail. A novel alias-suppression method was also described, and the various methods of timbre modification via low-frequency modulation were discussed. In a complementary manner, we also looked at audio-rate shape modulation synthesis and discussed the general principles of the technique. It is expected that the ideas proposed in this paper will find a good range of applications in musical sound synthesis. Sound examples and software are available at <http://www.acoustics.hut.fi/go/dafx11-vps>.

7. ACKNOWLEDGMENTS

The authors would like to acknowledge the support from the Academy of Finland (project no. 122815) which supported part of this research.

8. REFERENCES

- [1] J.-C. Risset, "An introductory catalogue of computer-synthesized sounds," Tech. Rep., Bell Telephone Labs., 1969.
- [2] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *J. Audio Eng. Soc.*, vol. 21, no. 7, pp. 526–534, 1973.
- [3] J. A. Moorer, "The synthesis of complex audio spectra by means of discrete summation formulas," *J. Audio Eng. Soc.*, vol. 24, no. 9, pp. 717–727, 1976.
- [4] D. Arfib, "Digital synthesis of complex spectra by means of multiplication of nonlinear distorted sine waves," *J. Audio Eng. Soc.*, vol. 27, no. 10, pp. 757–768, 1979.
- [5] M. Le Brun, "Digital waveshaping synthesis," *J. Audio Eng. Soc.*, vol. 27, no. 4, pp. 250–266, 1979.
- [6] J.-P. Palamin, P. Palamin, and A. Ronveaux, "A method of generating and controlling musical asymmetrical spectra," *J. Audio Eng. Soc.*, vol. 36, no. 9, pp. 671–685, 1988.
- [7] M. Puckette, "Formant-based audio synthesis using nonlinear distortion," *J. Audio Eng. Soc.*, vol. 43, no. 1/2, pp. 40–47, 1995.
- [8] V. Lazzarini, J. Timoney, and T. Lysaght, "The generation of natural-synthetic spectra by means of adaptive frequency modulation," *Computer Music J.*, vol. 32, no. 2, pp. 9–22, 2008.
- [9] V. Lazzarini and J. Timoney, "Asymmetric-spectra methods of adaptive FM synthesis," in *Proc. Digital Audio Effects (DAFx-08)*, Espoo, Finland, August 2008, pp. 233–240.
- [10] V. Lazzarini, J. Timoney, J. Pekonen, and V. Välimäki, "Adaptive phase distortion synthesis," in *Proc. Digital Audio Effects (DAFx-09)*, Como, Italy, September 2009.
- [11] V. Lazzarini and J. Timoney, "Theory and practice of modified frequency modulation synthesis," *J. Audio Eng. Soc.*, vol. 58, no. 6, pp. 459–471, 2010.
- [12] V. Lazzarini, J. Kleimola, V. Välimäki, and J. Timoney, "Five variations on a feedback theme," in *Proc. Digital Audio Effects (DAFx-09)*, Como, Italy, September 2009.
- [13] J. Kleimola, V. Lazzarini, J. Timoney, and V. Välimäki, "Feedback amplitude modulation synthesis," *EURASIP J. Advances in Signal Processing (JASP)*, vol. 2011, pp. 1–18, 2011.
- [14] M. Ishibashi, "Electronic musical instrument," U.S. Patent 4,658,691, 1987.
- [15] V. Lazzarini and J. Timoney, "New perspectives on distortion synthesis for virtual analog oscillators," *Computer Music J.*, vol. 34, no. 1, pp. 28–40, 2010.
- [16] J. Kleimola, V. Lazzarini, J. Timoney, and V. Välimäki, "Phaseshaping oscillator algorithms for digital sound synthesis," in *Proc. Sound and Music Computing Conf. (SMC 2010)*, Barcelona, Spain, July 2010.
- [17] J. Timoney, V. Lazzarini, A. Gibney, and J. Pekonen, "Digital emulation of distortion by wave and phase shaping methods," in *Proc. Digital Audio Effects (DAFx-10)*, Graz, Austria, September 2010.
- [18] V. Lazzarini and J. Timoney, "New methods of formant analysis-synthesis for musical applications," in *Proc. Intl. Computer Music Conf.*, Montreal, Canada, September 2009.
- [19] M. Le Brun, "A derivation of the spectrum of FM with a complex modulating wave," *Computer Music J.*, vol. 1, no. 4, pp. 51–52, 1977.
- [20] M. Corrington, "Variation in bandwidth with modulation index in frequency modulation," *Proc. IRE*, vol. 35, no. 10, pp. 1013–1020, 1947.
- [21] V. Verfaillie, U. Zölzer, and D. Arfib, "Adaptive digital audio effects (a-DAFx): a new class of sound transformations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1817–1831, Sept. 2006.
- [22] E. Maor, "Trigonometric delights," Princeton Univ. Press, Princeton, NJ, 1998.

NON-PARALLEL SINGING-VOICE CONVERSION BY PHONEME-BASED MAPPING AND COVARIANCE APPROXIMATION

Fernando Villavicencio and Hideki Kenmochi

Corporate Research
Development Center
Yamaha Corporation
Hamamatsu, Shizuoka, Japan
{villavicencio, kenmochi}@beat.yamaha.co.jp

ABSTRACT

In this work we present an approach to perform voice timbre conversion from unpaired data. Voice Conversion strategies are commonly restricted to the use of parallel speech corpora. Our proposition is based on two main concepts: the modeling of the timbre space based on phonetic information and a simple approximation of the cross-covariance of source-target features. The experimental results based on the mentioned strategy in singing-voice data of the VOCALOID synthesizer showed a conversion performance comparable to that obtained by Maximum-Likelihood, thereby allowing us to achieve singer-timbre conversion from real singing performances.

1. INTRODUCTION

One of the main limitations of current Voice Conversion technologies are the use of a *parallel* corpora of the source and the target speakers to perform training of a conversion model. This corpus consists of a set of recordings in which both speakers pronounce the same utterances (same phonetic content) without applying any distinctive emotion or vocal quality. The acquisition of such parallel data may represent a number of difficulties, especially if aiming to apply it on target *voices* which are hardly available; for example: past celebrities.

Some proposals have been reported to achieve non-parallel conversion based on the alignment of originally unpaired data by exhaustive similarity search or the adaptation of an original parallel model. An approach following the latter concept [1], is based on the assumption of a linear relation between the timbre features of originally *paired* speakers and *unpaired* ones. A conversion model trained from paired data is adapted accordingly; however, the source-to-target mapping is not defined directly from the unpaired data.

Previously, the authors introduced a strategy to derive the timbre conversion model exclusively from unpaired data considering that the phonetic segmentation is available [2]. The proposition consists of a modification of the original Gaussian Mixture Model (GMM) based approach of [3] and [4] by applying phoneme-constrained modeling of the timbre space and an approximation of the joint-statistics following the same assumption considered in [1]. In terms of spectral conversion error, the conversion performance was found comparable to that obtained by parallel training without perceiving a significant reduction of the conversion effect on the converted signals.

In this work we extend the study of the proposition presented in [2]. In particular, we are interested in clarifying issues as the learning conditions of the phoneme-constrained modeling and the performance of the proposed non-parallel approach when the nature of the source target corpora differs. We remark on our interest in applying this technology to the concatenative singing-voice synthesizer VOCALOID [5] in order to perform singer-timbre conversion on the system databases by exclusively using real performances from target singers. According to the work presented in [6], the experimental study was carried out on full-quality singing-voice data ($Sr = 44.1KHz$). However, the proposal presented in this work may represent a generalized solution for Voice Conversion purposes.

This paper is structured as follows: the phoneme-constrained Multi Gaussian Modeling is presented in section 2, in section 3 we show study of simple strategy to approximate the source-target cross-covariance, the experimental framework of our study is described in section 4, to evaluate the performance of the proposed method and compare it with the one based on ML, the results of objective and subjective evaluations are reported and discussed in section 5, and the paper concludes with observations and proposition for further study in section 6.

2. PHONEME-BASED ENVELOPE MAPPING

2.1. GMM-ML for features conversion

The conversion of the voice timbre is commonly achieved by modification of the short-term spectral envelope information based on a probabilistic time-continuous transformation function [3]. The conversion function is commonly derived from a Gaussian Mixture Model of joint timbre features trained in a ML basis. The timbre features correspond to all-pole based estimations of the spectral envelope parameterized as Line Spectral Frequencies (LSF) [4]. We remind, for clarity, the main expressions followed on this strategy

$$\hat{y} = \sum_{q=1}^Q p(q|x) [\mu_q^y + \Sigma_q^{yx} \Sigma_q^{xx-1} (x - \mu_q^x)] \quad (1)$$

$$p(q|x) = \frac{\mathcal{N}(x; \mu_q^x; \Sigma_q^{xx})}{\sum_{q=1}^Q \mathcal{N}(x; \mu_q^x; \Sigma_q^{xx})} \quad (2)$$

Eq.1 depicts the conversion function, denoting x, y and \hat{y} the source, target and converted envelope features respectively. The GMM size (number of Gaussian components) is given by Q . Note

that an *a priori* weighting of the mixture components is not considered. The term $p(q|x)$ corresponds to the conditional probability or *class membership*, according to Eq.2.

In general, concerning the configuration of the GMM, the number of Gaussian components depends on the amount of training data as well as the form of the covariance matrices (full or diagonal). Normally, an eight-sized GMM with full covariance matrices is in use to achieve learning generalization for voice conversion purposes [3]. Commonly, the resulting Gaussian means, when translated to the spectral domain, depict spectral envelopes with formantic features. This is principally due to the restriction of using only voiced speech and the significant amount of vocalic content on the data. Note, however, that a one-to-one correspondence cannot be straightforwardly stated between those envelope patterns and the vocalic phonetic classes assumed to be contained in the data.

The vocalic speech is widely considered as provider for the most important perceptual cues for timbre identification. However, they represent only a subset of the phonetic elements of a language. Subsequently, we claim that if aiming to perform full timbre conversion we might map the envelope characteristics regardless, in general, of their vocalic or voiced nature. Accordingly, a clustering of the envelope space by only eight gaussian components may lead to a large averaging of the phonetic content. Note also the highly competitive behavior observed on the ML-based mixture, resulting in a full modeling of speech segments of different phonetic nature by the same Gaussian distribution. These phenomena lead to a significant simplification of the phonetic space on the mapping process and are found at the origin of some “reduction” or modification of the phonetic content perceived in some converted utterances.

2.2. Phoneme-constrained Multi-Gaussian Model

Moreover, by means of setting the GMM size close to the assumed number of phonetic events and restricting the covariance matrices to be diagonal, the behavior of the mixture was found to be more cooperative but unstable. We show in Fig.1 the resulting component-to-phoneme correspondence for a GMM-ML in terms of the average membership of each gaussian per phonetic class. The results were obtained by evaluating $p(q|x)$ after training the GMM with labeled data. The vertical axis represents the GMM components whereas the horizontal axis lists the phonemes included in VOCALOID according to the Japanese language (SAMPA standard) ordered by phonetic group (vowels, nasals, voiced plosives, voiced affricates, liquids, semivowels, unvoiced plosives, fricatives, unvoiced affricates).

Clearly, following Fig.1, relationships between the clustering achieved by the GMM-ML and the phonetic class of the features can hardly be established. An unstable activation of the mixture components along with phonetic content may produce irregular evolution of the converted envelopes, representing a potential factor of degradations on the converted signals.

Consequently, we propose to control the fitting of the statistical model by using the phonetic information; therefore, we restrict the computation of each Gaussian distribution to the data corresponding to a same phoneme. A phoneme-based modeling (pho-GMM) was already introduced in [7], showing some benefits in terms of one-to-many mapping reduction compared to conventional GMM-ML.

Following this strategy the resulting component-to-phoneme

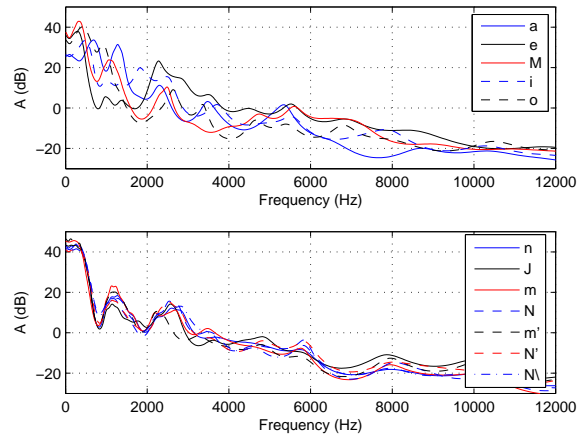


Figure 3: Corresponding spectral envelopes of the MGM means within phonetic groups. Vowels (top), nasals (bottom).

correspondence is clearly increased [2], as shown in Fig.2. The model was therefore able to extract characteristic information for most of the phonemes, and to increase, consequently, the discrimination between them.

Note however some “shared” regions on the grid within elements of a same phonetic group (e.g. nasals, plosives). Unlike the case of the vowels, where the differences between the formantic structures represent an important discriminant factor, the average spectral patterns at these groups are relatively close. This can be appreciated in Fig.3, where are shown the resulting envelope patterns of the vowels (top) and nasals (bottom) sets. Although we have not a theoretical basis to explain these similarities, a further simplification of the phonetic space and the role of such a “characteristic envelope” on non-stationary phonemes (e.g. plosives) may be studied.

Finally, keeping consideration that the phonetic information is available, the conditional probability can be replaced by a phonetic flag to directly assign the corresponding component at the conversion stage. However, this “forced” membership should be smoothed at the phonetic boundaries to avoid abrupt changes when transforming the signal. As was already described, by forcing a full-competitive behavior we do not significantly differ from the real role of $p(q|x)$ observed in a GMM-ML. Moreover, following this proposition we aim to refine the envelope mapping in a phonetic basis. Note however that, as commented in [7], without including more meaningful context information some mapping losses can be hardly alleviated if the acoustic characteristics of same-phoneme data significantly differs. This is demonstrated further in our experimentation by using data of increasing heterogeneity.

Accordingly, the original conversion function expressed in Eq. 1 is modified as

$$\hat{y} = \mu_{q(x)}^y + \sum_{q(x)}^{y,x} \Sigma_{q(x)}^{x,x}{}^{-1} [x - \mu_{q(x)}^x] \quad (3)$$

Moreover, the sub-index $q(x)$, denotes the phonetic class of the source input and therefore, defines the only gaussian component involved in the mapping. Subsequently, since the resulting model does not keep the “mixture” characteristic anymore, we refer to it as a “Multi-Gaussian Model” (MGM) [2].

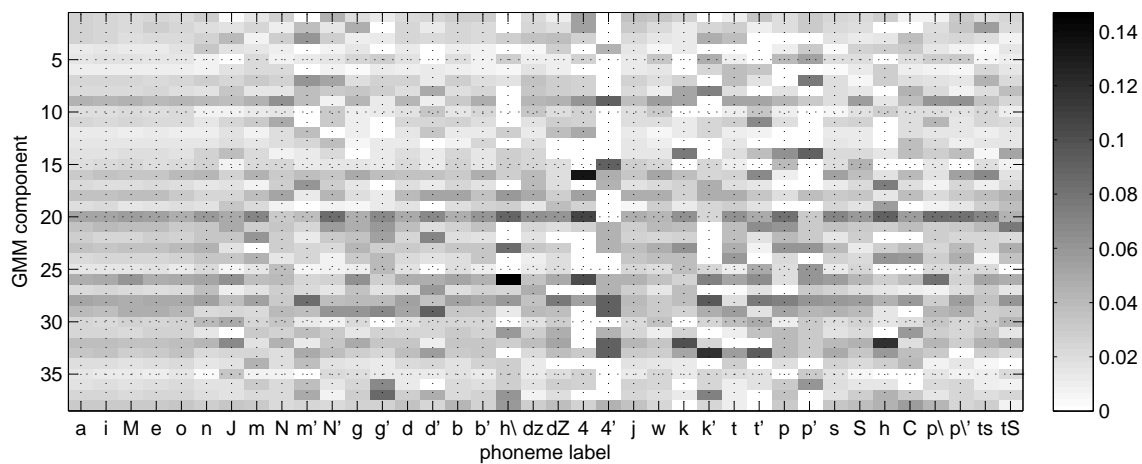


Figure 1: Average conditional probability at each GMM component per phonetic class. ML-based fitting.

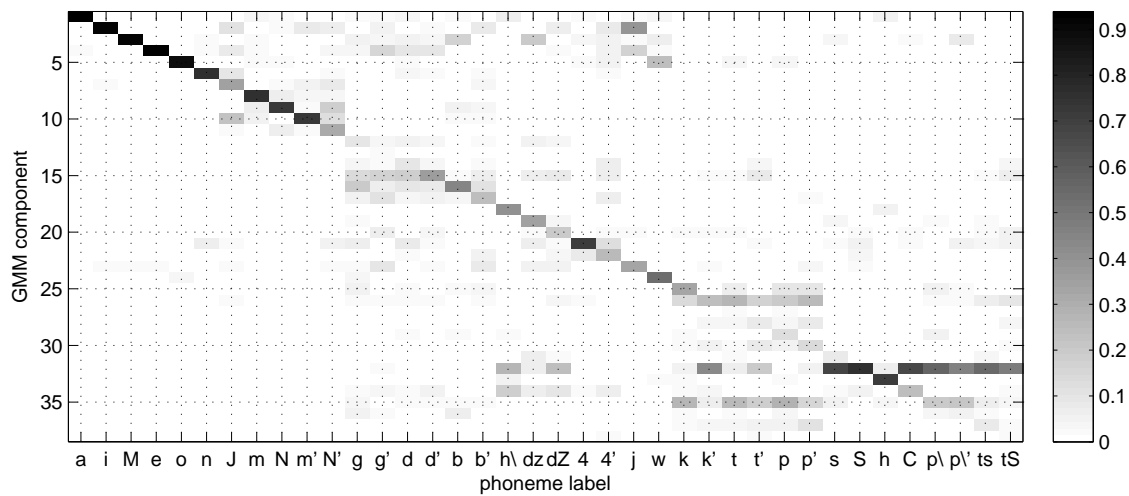


Figure 2: Average conditional probability at each GMM component per phonetic class. Phoneme-based fitting.

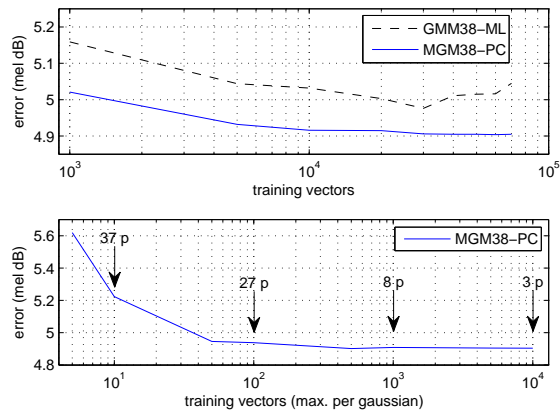


Figure 4: GMM-ML and pho-MGM conversion error by overall training size (top). pho-MGM Error by maximum training size per component (bottom). The error measure (mel dB) corresponds to the spectral distortion averaged over mel-scaled spectra.

2.3. MGM performance and training

We intended to study the minimal amount of data per phoneme required to generalize the timbre mapping. However, the amount of frames per phoneme can barely be equilibrated since the vocalic content is predominant in the Japanese language. Thus, we limit our data size control to an upper bound, or maximal training size, for the number of frames of a same phoneme used to fit a Gaussian component. A regularization of the covariance matrices was required for phonemes from which only a small amount of data was available.

The results are shown in Fig. 4. The timbre features correspond to LSF parameters of accurate spectral envelope estimates obtained by a mel-based autoregressive model [6] with envelope order set to 50. The cost function corresponds to the spectral conversion error between the converted envelopes and the target spectra. We compared GMM-ML and MGM models with similar complexity (diagonal matrices, 38 components). In general, the resulting conversion error levels are similar (top graph), showing the MGM with slightly increased performance. An over-fitting effect was not found to be affecting, though a small training set was used (1000 vectors). We remark that the conversion performance was always evaluated on unknown data (test set) in order to observe the stabilization of the learning; that explains the decreasing behavior of the error curve.

The maximal amount of training data per MGM component was also evaluated (bottom graph). The arrows denote the number of phonetic classes reaching the corresponding maximal number of vectors at each case. The results show that it is not necessary to have a large amount of frames (around 100) to approach the high performance region.

3. CROSS-COVARIANCE APPROXIMATION

3.1. Motivation

From eq. 3 we remark that the only term for which paired data is required is the source-target cross-covariance (Σ^{yx}). By simplify-

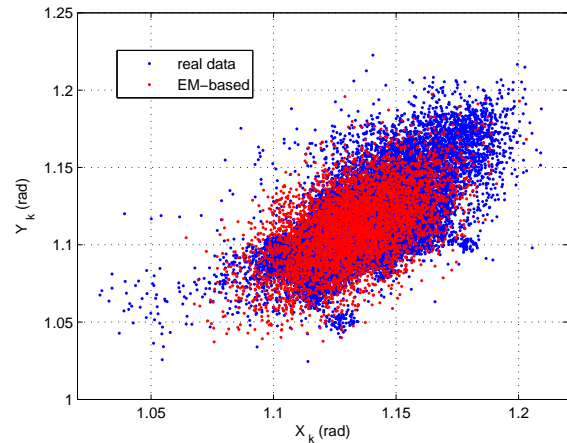


Figure 5: Example of LSF data within a phoneme-class (one-dimension). Real data (blue) and generated from the resulting ML-based Gaussian distribution (red).

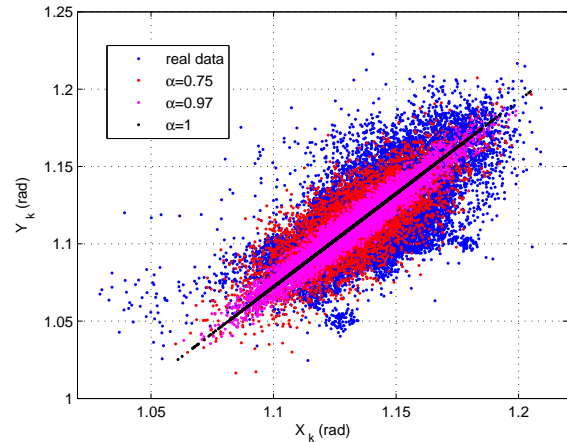


Figure 6: Example of LSF data within a phoneme-class (one-dimension). Real data (blue) and generated from the approximated statistics for variable α (black, magenta, red).

ing the proposition of [1] via assuming directly a linear transformation between the source and target features their joint statistics can be approximated. Moreover, the phoneme-constrained modeling presented in the past section limits this term, for each Gaussian distribution, to depend exclusively on the data of the corresponding phonetic class.

According to eq. 3, the term Σ^{yx} , commonly called *transformation matrix* after normalization by the source variance, acts actually as a weight of the variance of the converted features. The values observed on this term on the GMM-ML based models are rather small, resulting in poor dynamics of the converted features. This well-known and characteristic over-smoothing, already addressed in works [8], is commonly perceived as a *muffling* quality, affecting the naturalness of the converted signals.

Notably, an augmentation of the variance of the oversmoothed converted parameters has been found to reduce significantly this

muffling effect. Therefore, we assert that this term when estimated by ML represents a limitation on the resulting conversion quality. Furthermore, having control of this value might represent an effective way to increase the naturalness of the converted signals.

3.2. Covariance approximation by linear transformation

Following the phoneme-constrained modeling, the probabilistic linear transformation between the timbre features of two speakers proposed in [1] can be simplified as $y = A_{q(x)}x + b_{q(x)}$, where A_q is, in general, a square matrix according to the dimensionality of x , and b_q is a bias vector. Therefore, considering the mentioned relation in the computation of Σ^{yx} for each phonetic-component of the MGM we obtain

$$\tilde{\Sigma}^{yx} = E[(\tilde{y} - \mu^{\tilde{y}})(x - \mu^x)] \quad (4)$$

$$= E\{[(Ax + b) - (A\mu^x + b)](x - \mu^x)\} \quad (5)$$

$$= E[(A(x - \mu^x))^2] = A\Sigma^{xx} \quad (6)$$

Where A can be approximated similarly by evaluating Σ^{yy}

$$\tilde{\Sigma}^{yy} = E\{[(Ax + b) - (A\mu^x + b)]^2\} \quad (7)$$

$$= E[(A^2(x - \mu^x)^2)] = A^2\Sigma^{xx} \quad (8)$$

$$A = \sqrt{\Sigma^{yy}\Sigma^{xx-1}} \quad (9)$$

Although the relation $y = Ax + b$ is assumed between features corresponding to the same phoneme imposes a strong assumption and, by using diagonal covariance matrices, the resulting one-dimensional distributions restricts to narrow regions. As the norm of A decreases, the “width” of the covariance region increases until it reaches a circular form at the full-uncorrelated case ($A = 0$). Thus, since the orientation of the modeled distribution is given exclusively by Σ^{xx} and Σ^{yy} the proposed $\tilde{\Sigma}^{yx}$ may be rather seen as a lower bound of the real distribution width. Accordingly, we apply a weighting factor ($0 < \alpha < 1$) to $\tilde{\Sigma}^{yx}$ on the conversion function in order to impose a more realistic form on the approximated distribution.

In Fig. 6 we show a comparison of real and approximated source-target distributions for several α values of one LSF dimension within a phonetic class. Clearly, the distribution strictly following the relation $y = Ax + b$ ($\alpha = 1$) does not suffice the data. However, by setting α around 0.75, the covariance region approaches the covariance based on ML. This can be seen in Fig. 5, illustrating the case when the Gaussian is fitted in a ML basis.

Then, based on eq. 3, the final expression for the conversion features will be as follows

$$\hat{y} = \mu_{q(x)}^y + \alpha \sqrt{\Sigma_{q(x)}^{yy}\Sigma_{q(x)}^{xx-1}} [x - \mu_{q(x)}^x] \quad (10)$$

Regarding the effect of the parameter α on the conversion performance, values within the range [0.5-0.7] provide the best perceptual results. Further, we observe that, for the dimensions with a low correlation, the imposition of a covariance value higher than the real one was found to be beneficial. The naturalness of the converted signals is improved by increasing the dynamics of the predicted LSFs. Nevertheless, for clarity an objective and subjective evaluation of α is presented in Section 5.

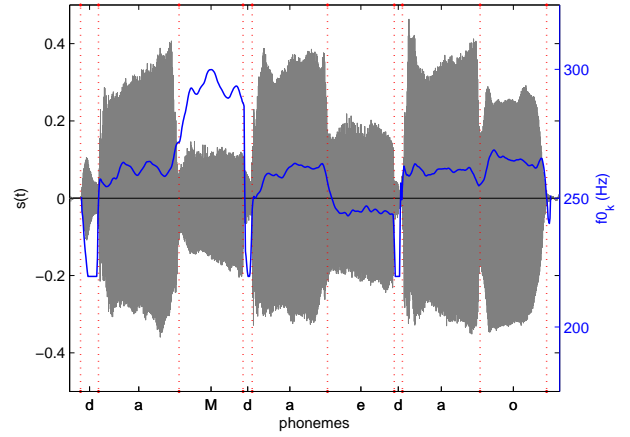


Figure 7: Example of a singing sample of a VOCALOID DB including the phonetic segmentation and F0 estimation.

4. EXPERIMENTAL FRAMEWORK

4.1. VOCALOID singer databases

The VOCALOID singing-voice synthesizer system consists of three main elements, the user interface (allowing to input lyrics and melody information), a singer database (containing a collection of singing-voice samples from a singer), and the synthesis engine (performing, briefly, the selection, F0-transposition and concatenation of the samples).

In particular, the singer-DB consists of a pre-defined set of phonetic sequences sung at different pitch ranges. The phonetic scripts are assumed to cover principally the consonant-vowel combinations of the Japanese language. All the singing samples are recorded at a same tempo following the same melodic pattern, which is restricted to a one tone variation related to the representative musical height of each pitch set. An example of a singing sample is shown in Fig. 7. Each single-pitch set consists of more than 100 recordings, representing more than 70,000 feature vectors. Typically, a complete VOCALOID-DB includes low, medium, and a high pitch sets.

Singing-voice data from 2 VOCALOID singer-DBs were considered for our experimental study. A C4 (261Hz) pitch set of a female singer was set as source voice whereas G3, C4 and E4 pitch sets (193, 261, 330Hz) as well as 4 real singing performances from a male singer were used as target voice. The configuration of the target data was modified according to the interest of comparing both the mapping strategy and the effect of the pitch on the conversion performance, and is described in the next section.

4.2. Effect of the corpora heterogeneity

There is advantage, in terms of envelope mapping performance, to using data which is not only paired but restricted to a small pitch variation [6]. Accordingly, the use of non-parallel corpora may have an impact on the conversion performance if the nature of the source and target corpora differs. We remark that one of our main interests in applying non-parallel timbre conversion on the singing-voice is to use real singing performances to compute the

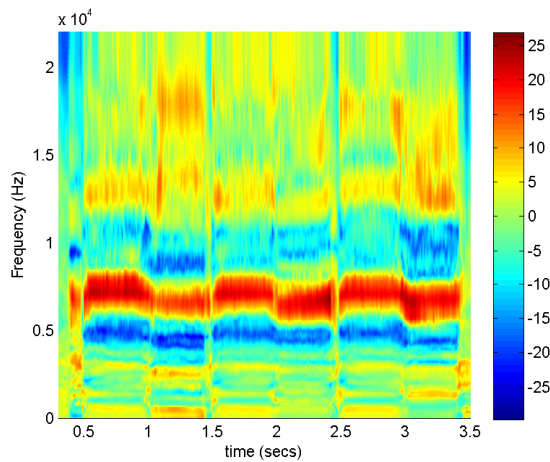


Figure 8: Spectrogram of the conversion filter for the sample of Fig. 7 (energy in dBs).

conversion model, which may observe a large pitch range (melody), different tempo, and a rich pronunciation variety among the content.

We were therefore interested in studying the performance using data with different characteristics. Therefore, we fixed three different sets as target data: a) VOCALOID's single-pitch data, b) VOCALOID's mixed-pitch data and c) real singing performances (4 songs). Further, we considered the following evaluation cases: a) GMM-ML single-pitch (labeled as GMM-ML P SP), b) MGM single-pitch (MGM-PC P SP), c) non-parallel single-pitch (MGM-PC NP SP), d) non-parallel mixed-pitch (MGM-PC NP MP), and e) non-parallel real songs performances (MGM-PC NP RP). Since objective evaluation on unpaired data is not straightforward, all the approaches were evaluated on the single-pitch paired data i.e., different sets for training but same ones for evaluation.

An evaluation of this nature represents the most exigent case since the single-pitch set observes the most precise and "homogeneous" features, resulting in an increased challenge for the models trained on data corresponding to wider pitch-range and "heterogeneous" phonation characteristics (multi-pitch and real singing performances sets).

4.3. Signal modification

As was already described, the timbre conversion process is based in a short-term mapping of the spectral envelope information. The transformation of the timbre is therefore achieved by replacing the original envelope by the one given by the converted features. This is commonly done by analysis-synthesis filtering following the autoregressive modeling of the envelope. However, we perform the modification of the envelope by defining a *conversion filter*, corresponding to the difference at each frame between the corresponding transfer function of the converted LSFs and an interpolation of a harmonic analysis of the source signal. The frame processing is done in a pitch-synchronous basis. We show in Fig. 8, in the form of a spectrogram, an example of resulting conversion filter for the utterance of Fig. 7.

We use the harmonic information instead of the envelope on

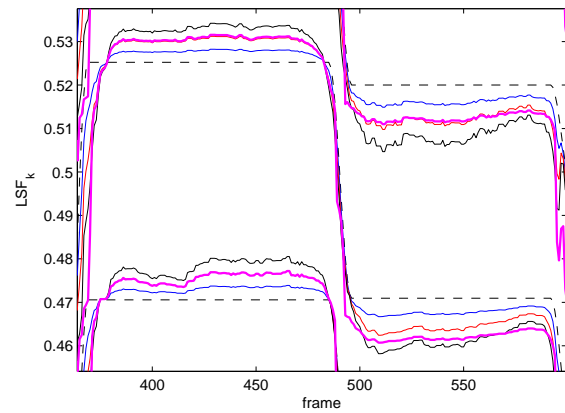


Figure 9: Converted LSF parameters given by $\alpha = 0$ (dotted), $\alpha = 0.25$ (blue), $\alpha = 0.5$ (red), $\alpha = 0.75$ (black) and ML-based conversion (magenta).

the source signal aiming to match closely the real source information and discard the risk of estimation errors that occurred during the computation of the autoregressive model. The harmonic analysis and the processing framework itself follow the wide-band technique described in [9].

Besides the capability of the processing method to perform efficient envelope modification, the conversion itself may result in some unnatural or distorted quality on the transformed signals since the characteristics of the converted spectra may not match naturally the original signal (harmonicity, energy, f_0). Also, consider that some abrupt changes on the evolution of the source signal cannot be properly reflected by the mapping process.

Moreover, the stable and controlled characteristics of the singing samples might impact positively the conversion quality if compared to the case of spontaneous speech. However, the particular evolution of the source signal and the use of wide-band based information may result in an important variation of the envelope information at successive frames. Accordingly, we consider two parameters to control independently the smoothness of the conversion filter for both time and frequency axes. Although it is not generally required, this strategy was found effective to avoid degradations in some converted utterances and to smooth undesired frame-to-frame variations on the conversion filter.

5. EVALUATION

5.1. Objective evaluation

We were interested on studying three aspects in our experimental evaluation: first, the impact of the covariance approximation on the converted features, second, to compare the conversion performance of the parallel and non-parallel strategies, and finally to evaluate the effect of the heterogeneity of the target data.

We therefore started analyzing the converted LSFs for different α values. Note the benefits of using this parameterization, seen as temporal trajectories denoting spectral pole locations, to observe differences in terms of variance. This can be seen in Fig. 9. The plot shows a comparison of three converted LSFs at a segment

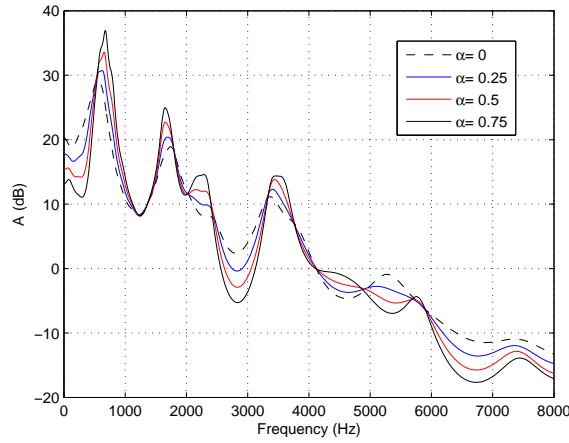


Figure 10: Resulting spectral envelopes from converted LSF parameters for different α values.

of the utterance example. The different cases correspond to conversions obtained by using increasing α values as well as a ML result issued from a model with similar complexity. As expected, an augmentation of this value was found to increase the temporal variance related to the means position ($\alpha = 0$). The corresponding effect in the spectrum is found as an emphasis of the maxima and minima of energy, as shown in Fig. 10, producing a positive effect, within a reasonable limit, in the perceived naturalness.

Fig. 11 shows the average variance measured on about 5000 evaluation vectors (test set) of target and predicted LSF for the different evaluation cases as described in section 4.2. Note that the resulting variances of the parallel cases are just slightly higher than those given by the gaussian means ($\alpha = 0$), denoting the poor impact of the transformation matrix when it is exclusively derived from the data (whether or not the variance is obtained by ML). On the other hand, note that by setting α we can force a variance on the converted features close to the real one.

Finally, Fig. 12 depicts a conversion performance comparison. The cases involving models trained on single-pitch data (labels ending with “SP”) are considered as the references since the training data corresponds to similar corpora with stable characteristics. The proposed non-parallel conversion using single-pitch data performs close of ML and MGM-parallel cases. As expected, the performance decreases as the target data is more heterogeneous. Moreover, the conversion performance shows a maximum related to α ; however, slightly higher values ([0.5-0.7]) have been found as providing increased naturalness (muffled quality reduction).

5.2. Subjective evaluation

A subjective evaluation was designed aiming to compare the conversion effect and the quality of the converted signals. Looking for a strict perceptual evaluation, ten sound technology professionals participated as listeners. Five VOCALOID samples, representative of the different phonetic groups, were selected as the evaluation set.

First, a timbre similarity test was defined considering three conversion strategies: GMM ML SP, MGM NP SP and MGM

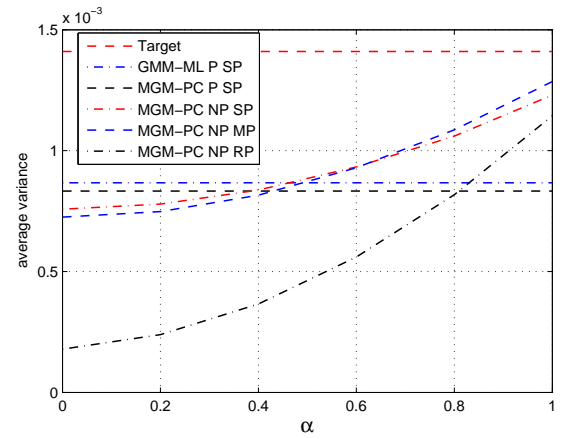


Figure 11: Average variance of target and converted LSFs for the different method and data configurations.

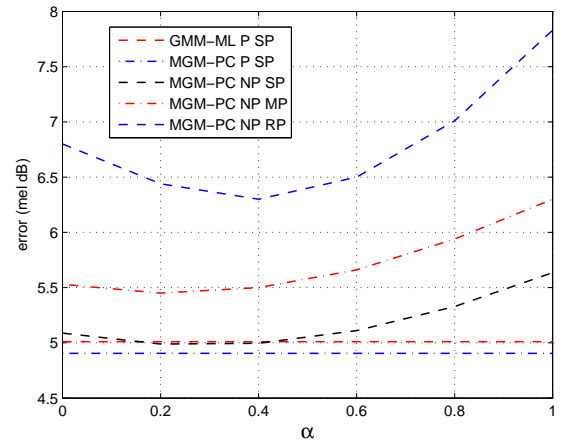


Figure 12: Spectral conversion error for the different method and data configurations.

NP RP. We intended to compare the conversion effect when using both parallel and non-parallel methods and the effect of using a homogenous or heterogeneous target corpus on the proposed non-parallel method. The procedure was as follows: the source, target, and the three converted utterances (randomly selected) were presented to the listeners. Then, the timbre similarity between the converted and the reference samples was measured according to the continuous range [0 1] (0 = source, 1 = target). This process was repeated immediately to allow confirmation or modification of the first judgement.

Note that at each sample case, both reference and converted utterances observe stable and similar acoustic characteristics (pitch, energy, and dynamics). A comparison based on this data appears to be an efficient way to exclusively focus on the timbre and vocal quality differences. However, the conversion of vocal quality features is out of the scope of this work. We claim that although an increased perceptual discrimination capacity may result in lower

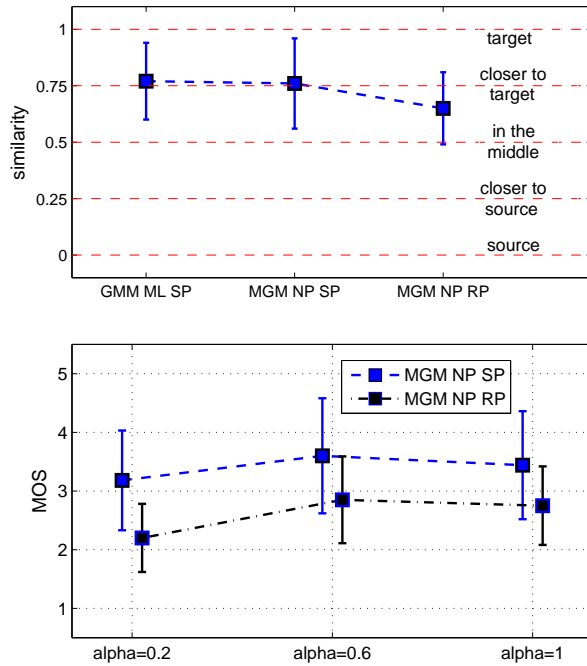


Figure 13: Subjective evaluation. Timbre similarity results (top) according to given reference levels. Signal quality evaluation (MOS) of the non-parallel conversion for different α values.

conversion scores it might lead us to a more robust evaluation of the effect achieved by the spectral envelope conversion process.

The results are shown in Fig. 13 (top). In the figure the five levels in red tagged with a subjective description correspond to the scale references given to the listeners. The scores achieved by both parallel and non-parallel methods when using the same corpora were found similar and denote, in general, a reasonable conversion effect. However, the performance suffers some reduction when real singing corpora is used as target training data.

Second, a MOS-like test was focused on exclusively evaluating the signal quality of the non-parallel conversion for different α values. The test was applied separately for both corpora cases in order to observe exclusively the effect of α . The subjective description of the MOS levels was re-defined looking for an exigent evaluation and an association of the measurement levels with some quality phenomena (5=perfect, 4=slight degradations, 3=clean enough, 2=artifact(s), 1=annoying).

The test followed a similar procedure as the similarity test. For each sample case three converted samples, corresponding to three representative α values (small, $\alpha=0.2$; proposed, $\alpha=0.6$; large, $\alpha=1$), were randomly selected and evaluated in the same two-step basis. The results are shown in Fig. 13 (bottom). As expected, best results were found for $\alpha=0.6$. Note however that a large value achieved a comparable performance. This might be explained by a reduced risk of producing undesired amplitude modulations on the spectrum when applying a high variance to the LSF trajectories on stable signals.

Although the overall results does not allow us to claim full natural-quality conversion the scores achieved when using similar

and stable corpora show a general perception of an adequate naturalness. As for the similarity test, the drop in the performance level is attributed to the increased heterogeneity of the target corpora, resulting in over-smoothed envelope patterns on the conversion model. The estimation of precise envelope information from singing performances might be studied further.

6. CONCLUSIONS AND FUTURE WORK

In this work we presented an approach to perform voice timbre-conversion from non-parallel data. The proposed strategy is based on phoneme-constrained modeling of the statistical space of the timbre features and an approximation of the cross-covariance information and is described and compared with the conventional approach based on parallel data and ML. The results, obtained from an experimental study on singing-voice let us claim the achievement of comparable conversion performance although some dependency was observed according to the heterogeneity of the corpora.

The experimentation done in this work suggest to extend the study in some issues: the estimation of the α parameter individually for each feature dimension; an efficient selection of envelope features from real singing performances; An efficient mapping of non-stationary phonemes, among others. However, the proposition presented in this work was proved to be a step-forward the interests of voice timbre conversion.

7. REFERENCES

- [1] A. Mouchtaris, J. Van der Spiegel, and P. Mueller, "Non-parallel training for voice conversion based on a parameter adaptation approach," *IEEE-TASLP*, vol. 14, no. 2, pp. 952–963, 2006.
- [2] F. Villavicencio and H. Kenmochi, "Resurrecting past singers: Non-parallel singing-voice conversion," in *11th International Workshop on Singing-Voice InterSinging*, 2010.
- [3] Y. Stylianou, O. Cappé, and E. Moulines, "Continuous probabilistic transform for voice conversion," *IEEE-TASAP*, vol. 6, no. 2, pp. 131–142, 1998.
- [4] A. Kain, *High-Resolution Voice Transformation*, Phd. thesis, Oregon Institute of Science and Technology, October 2001.
- [5] H. Kenmochi and H. Oshita, "Vocaloid commercial singing synthesizer based on sample concatenation," in *Proc. of INTERSPEECH'07*, 2007.
- [6] F. Villavicencio and J. Bonada, "Applying voice conversion to concatenative singing-voice synthesis," in *Proc. of INTERSPEECH'10*, 2010, vol. 1.
- [7] E. Godoy, O. Rosec, and X. Chonavel, "Alleviating the one-to-many mapping problem in voice conversion with context-dependent modeling," in *proc. of INTERSPEECH'09*, Brighton, UK., 2009.
- [8] T. Toda, A.W. Black, and Tokuda, "Spectral conversion based on maximum likelihood estimation considering global variance of converted parameter," in *Proceedings of IEEE-ICASSP '05*, 2005, pp. 1–9–1–12.
- [9] J. Bonada, "Wide-band harmonic sinusoidal modeling," in *International Conference on Digital Audio Effects, DAFx'08*, Helsinki, Finland, 2008.

APPLICATION OF NON-NEGATIVE MATRIX FACTORIZATION TO SIGNAL-ADAPTIVE AUDIO EFFECTS

Ryan Sarver

Centre for Digital Music
Queen Mary University of London, UK
rpsarver@gmail.com

Anssi Klapuri

Centre for Digital Music
Queen Mary University of London, UK
anssi.klapuri@eeecs.qmul.ac.uk

ABSTRACT

This paper proposes novel audio effects based on manipulating an audio signal in a representation domain provided by non-negative matrix factorization (NMF). Critical-band magnitude spectrograms \mathbf{Y} of sounds are first factorized into a product of two lower-rank matrices so that $\mathbf{Y} \approx \mathbf{B}\mathbf{G}$. The parameter matrices \mathbf{B} and \mathbf{G} are then processed in order to achieve the desired effect. Three classes of effects were investigated: 1) dynamic range compression (or expansion) of the component spectra or gains, 2) effects based on rank-ordering the components (columns of \mathbf{B} and the corresponding rows of \mathbf{G}) according to acoustic features extracted from them, and then weighting each component according to its rank, and 3) distortion effects based on controlling the amount of components (and thus the reconstruction error) in the above linear approximation. The subjective quality of the effects was assessed in a listening test.

1. INTRODUCTION

Audio effects can be viewed as processing modules that take in an audio signal and modify it according to certain control parameters to produce the desired audio output [1]. Typical examples include dynamic range compression, reverberation, and non-linear distortion for the electric guitar. The widespread use of audio effects in recorded music motivates the creation of new types of effects that produce musically interesting results and can be controlled by intuitive parameters.

During the last ten years, non-negative matrix factorization (NMF) has been actively studied for the purposes of audio content analysis [2, 3, 4, 5]. However, the potential of NMF for digital audio effects has not been properly investigated. NMF decomposes an input signal into a set of “components” that often correspond to physically distinct sources or sound events, and thereby opens a way towards applying effects on each source separately. For example, dynamic range compression can be applied on each component, instead of compressing the wideband signal or the signals within fixed subbands. In this paper, we propose three different strategies for manipulating an audio signal in the representation domain provided by the NMF before resynthesizing it back to a time-domain waveform. The results were evaluated in a listening test where the subjects described the differences they heard between the affected samples and the original ones and gave their opinions on whether the effect was interesting and useful. Overall, the results were positive and encourage further work in this area. Audio examples of the proposed effects are available at <http://www.elec.qmul.ac.uk/people/anssik/NMFEffects/>

2. METHOD

2.1. Data Representation

The effects discussed in this paper are based on factorizing the magnitude spectrograms of audio signals. The short-time Fourier transform (STFT) of a time-domain signal $x(n)$ is first calculated as

$$X_t(k) = \sum_{n=0}^{N-1} x(tH + n)w(n)e^{-j2\pi kn/N}, \quad (1)$$

where t is frame index, k is frequency index, N is the frame size, $H = N/2$ is the frame hop, and $w(n)$ is the hamming window.

The frequency resolution of STFT is linear, whereas the human auditory system carries out frequency analysis on a nonlinear scale. The equivalent rectangular bandwidths b_c of the critical bands in human hearing are given by [6]

$$b_c = 0.108f_c + 24.7 \text{ Hz}, \quad (2)$$

where f_c and b_c denote the center frequency and bandwidth of critical band (“channel”) c , and $c = 0, 1, \dots, C - 1$. The bandwidth b_c can be viewed as the frequency resolution of the peripheral auditory system at frequency f_c .

The perceptual quality of the audio effects obtained using NMF is greatly improved by warping the linear frequency resolution of the STFT to a critical-band resolution. This is achieved by simulating a bank of critical-band bandpass filters in the frequency domain. The center frequencies f_c of the filters that we use are distributed uniformly on the critical band scale (obtained by integrating the inverse of (2)),

$$f_c = 229 \left[10^{(a_1 c + a_0)/21.4} - 1 \right], \quad (3)$$

where $a_0 = 1.5$ determines the center frequency of the lowest band (40 Hz) and $a_1 = 0.79$ determines the band density in critical bandwidth units. We use a total of $C = 50$ subbands between 40 Hz and 20 kHz.

Warping from a linear frequency scale to the critical band scale is achieved using triangular sub-band responses (basis functions) that assign appropriately weighted STFT frequency bin values to the corresponding critical-band spectrogram bins. The basis functions are stored as rows in matrix \mathbf{W} which maps the STFT magnitude spectrogram $|\mathbf{X}|$ of size $(K \times T)$ to a critical-band spectrogram \mathbf{Y} of size $(C \times T)$ by

$$\mathbf{Y} = \mathbf{W}|\mathbf{X}|. \quad (4)$$

Figure 1 illustrates the structure of the basis matrix \mathbf{W} .

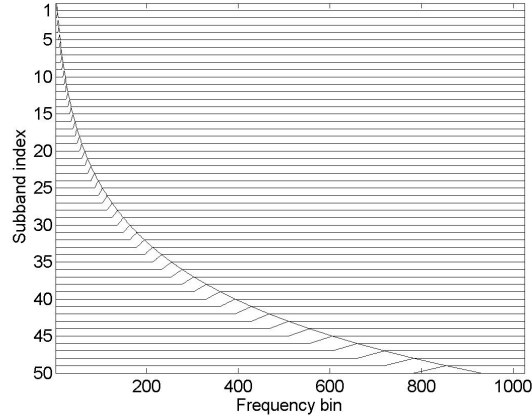


Figure 1: Illustration of the contents of the basis matrix \mathbf{W} used to warp from a linear frequency scale to a critical band scale.

2.2. Non-negative Matrix Factorization

The idea of NMF is to approximate a non-negative matrix $\mathbf{Y} \in \mathbb{R}_+^{C \times T}$ as a product of two lower-rank (that is, smaller) matrices $\mathbf{B} \in \mathbb{R}_+^{C \times Z}$ and $\mathbf{G} \in \mathbb{R}_+^{Z \times T}$:

$$\mathbf{Y} \approx \mathbf{B}\mathbf{G}. \quad (5)$$

The columns of matrix \mathbf{B} contain the spectra of individual components z , $z = 1, 2, \dots, Z$, and the rows of matrix \mathbf{G} contain the corresponding time-varying gains. The number of columns in \mathbf{B} (and rows in \mathbf{G}) is here denoted by Z and determines the number of components that \mathbf{Y} is broken into. Since magnitude spectra are inherently non-negative, a non-negativity restriction can be placed on these matrices [4]. Since the power spectra of many natural sounds (such as drum hits or individual notes) remains quite consistent across different occurrences, the factorization (5) often results in the separation of meaningful sound sources [4].

The algorithm we used for learning \mathbf{B} and \mathbf{G} is based on minimizing the Kullback-Leiber divergence between \mathbf{Y} and $\mathbf{B}\mathbf{G}$. The algorithm works by initializing \mathbf{B} and \mathbf{G} with random positive values and updating them iteratively with multiplicative rules until the algorithm converges [2]. The value of the cost function is decreased at each update until a local minimum is reached. The update rules for \mathbf{B} and \mathbf{G} are given by

$$\mathbf{B} \leftarrow \mathbf{B} \times \frac{(\mathbf{Y} ./ \mathbf{B}\mathbf{G}) \mathbf{G}^T}{\mathbf{1} \mathbf{G}^T} \quad (6)$$

$$\mathbf{G} \leftarrow \mathbf{G} \times \frac{\mathbf{B}^T (\mathbf{Y} ./ \mathbf{B}\mathbf{G})}{\mathbf{B}^T \mathbf{1}} \quad (7)$$

where $\mathbf{1}$ is a K -by- T matrix of ones, and \times and $./$ denote element-wise multiplication and division, respectively [2].

2.3. Resynthesis

The proposed audio effects are based on manipulating the matrices \mathbf{B} and \mathbf{G} before resynthesis. Before discussing the actual effects, however, let us consider the resynthesis of a time-domain signal from the NMF representation.

2.3.1. Direct Resynthesis from the Linear Model

The most straightforward way of resynthesis is based on the linear signal model of NMF directly:

$$\hat{\mathbf{Y}} = \mathbf{B}\mathbf{G} \quad (8)$$

This is followed by a warping of the critical-band scale back to the linear frequency scale, achieved using a transpose of the matrix of basis functions \mathbf{W} :

$$|\hat{\mathbf{X}}| = \mathbf{W}^T \hat{\mathbf{Y}} \quad (9)$$

The resulting magnitude spectrogram is combined with the phase spectrogram of the original mixture signal. Finally, inverse Fourier transform of each frame and 50% overlap-add is performed to obtain a time-domain signal.

2.3.2. Perfect Reconstruction Resynthesis

Synthesising a time-domain signal using (8) leads to inevitable distortion if the number of components Z is insufficient to represent the input audio spectrogram accurately. A typical requirement for audio effects is that the user can control the amount of effect applied on the input signal, and when this “effect depth” parameter is set to zero, the output signal is identical to the input signal (perfect reconstruction).

Perfect reconstruction resynthesis is achieved by reconstructing the complex-valued STFT spectrogram of component z by

$$\mathbf{X}_z = \left[\mathbf{W}^T \begin{pmatrix} \mathbf{b}_z \mathbf{g}_z \\ \mathbf{B}\mathbf{G} \end{pmatrix} \right] \times \mathbf{X} \quad (10)$$

where \mathbf{b}_i and \mathbf{g}_i denote the z th column of \mathbf{B} and the z th row of \mathbf{G} , respectively, and \mathbf{X} is the complex-valued STFT spectrogram of the input signal. This is one form of the Wiener filter and leads to perfect reconstruction of the complex-valued STFT spectrogram of the input signal by

$$\mathbf{X} = \sum_z \mathbf{X}_z \quad (11)$$

Inverse Fourier transform of \mathbf{X} followed by overlap-add can then be used to reconstruct the original input signal.

2.4. Audio Effects in the “NMF Domain”

The effects proposed in this paper are based on processing the parameter matrices \mathbf{B} and \mathbf{G} before resynthesizing the signal. For convenience in the following, we use the term “NMF domain” to refer to the parametric representation (5) of the input signal provided by the NMF.

2.4.1. Dynamic Range Compression and Expansion

Dynamic range compression and expansion involve multiplying the input signal by a slowly-varying gain factor that depends on the level of the input signal [7]. The operation of a dynamic range controller is typically described using a piece-wise linear curve that defines the desired output level (in decibels) as a function of the input level (in decibels). If the slope of this curve is $\frac{1}{3}$, for example, any change ΔL_i in the input level is mapped to a three times smaller change ΔL_o in the output level and the corresponding compression ratio $R = \Delta L_i / \Delta L_o$ would be 3. The term compression refers to $R > 1$ and expansion to $R < 1$.

A straightforward implementation of compression in the NMF domain can be achieved by raising the gains $g_z(t) \equiv \mathbf{g}_z$ of component z to power $1/R$. If the same amount of compression or expansion is to be applied on all components, then all elements of the matrix \mathbf{G} are raised to power $1/R$. For example, compression by factor 3 is achieved by raising all elements of \mathbf{G} to power $1/3$. This can be viewed as compression/expansion without a threshold (i.e., there is no threshold level below which the effect would be switched off).

Intuitively, compressing the component gains brings the less prominent sounds (at a given time) more to the foreground, since individual components tend to capture physical sound events or sound sources on the recording.

In the experiments to be described in Section 3, we investigated dynamics processing of not only the gain matrix but also the spectral basis matrix \mathbf{B} . Compression of the spectral basis matrix \mathbf{B} results in spectra of the individual components that are either compressed (flattened) or expanded.

2.4.2. Effects Based on Ordering the Components

The factorization achieved by NMF suffers from permutation ambiguity: the order of the components (columns of \mathbf{B} and the corresponding rows in \mathbf{G}) is arbitrary and depends on the random initialization of the matrices \mathbf{B} and \mathbf{G} before applying the multiplicative updates (6)-(7). In this sense, the individual components have no “identity”.

The class of effects described in this subsection is based on ordering the components $z = 1, 2, \dots, Z$ according to acoustic features calculated from the component spectra and gains. The components are then weighted differently before resynthesis, depending on their position on the ordered list.

Two different ordering criteria were investigated: spectral centroid and kurtosis. Spectral centroid is here defined as the first moment of the spectrum of a given component and conveys information about the “brightness” of that component. The spectral centroid S_z of component z is given by:

$$S_z = \frac{\sum_c f_c \mathbf{b}_z(c)}{\sum_c \mathbf{b}_z(c)} \quad (12)$$

where f_c is the center frequency of the critical band corresponding to bin c of matrix \mathbf{B} and is given by (3).

The spectral centroids S_z were then utilized to produce an audio effect by weighting component z by $(r_z - 1)/(Z - 1)$ before resynthesis. Here r_z denotes the rank of component z on a list where components are sorted in either ascending or descending-centroid order.

Another criterion that we used for ordering the components was the kurtosis. The kurtosis of the gain function $g_z(t)$ of component z is given by

$$K_z = \frac{\frac{1}{T} \sum_{t=1}^T (g_z(t) - \bar{g}_z)^4}{\left(\frac{1}{T} \sum_{t=1}^T (g_z(t) - \bar{g}_z)^2 \right)^2} - 3 \quad (13)$$

where \bar{g}_z denotes the empirical mean of $g_z(t)$. Note that the term -3 has no effect on the order and can be discarded.

We investigated ordering the components according to the kurtosis of their gains as well as the kurtosis of their spectra. Kurtosis of the gains function characterizes the “transientness” of a component (peakiness of its gains). Kurtosis of the spectrum, in turn,

tends to be higher for harmonic spectra (components representing musical notes) than for “noisy” spectra (components representing drum sounds for example). Similarly to the ordering based on spectral centroid, components were then scaled by a weight between 0 and 1 depending on their rank on the sorted list of components formed according to the kurtosis value.

2.4.3. Distortion as an Effect

The third class of effects is based on a controlled use of the reconstruction error caused by a direct resynthesis from the NMF model as described in Section 2.3.1. The distortion resulting from the NMF decomposition sometimes produces interesting effects in itself as will be discussed in the Results section. The effect was presented by cross-fading from the clean input signal to a signal reconstructed from an NMF model with eight components. This was then further cross-faded to a signal obtained using four, two, and finally just one component, and then back to the clean signal in the opposite order. The number of components used controls the amount of distortion introduced.

3. RESULTS

As the success of an audio effect cannot be assessed objectively, we conducted a listening test where the subjects rated and described the effects they heard. The current implementation of the method is non-causal and requires off-line processing of the input signals. Therefore the parameters of the effects in the listening test had to be fixed and the test stimuli calculated in advance, as opposed to allowing the subjects to tune the parameters in real-time. We chose parameters and music clips that were thought to be representative and interesting examples of each class of effects. The samples and the used parameter values are available on-line at <http://www.elec.qmul.ac.uk/people/anssik/NMFEffects/>

3.1. Stimuli

Four clips of music were chosen that were thought to best exemplify the investigated effects. The clips were from *Smells Like Teen Spirit* by Nirvana, *Billie Jean* by Michael Jackson, *Come Together* by the Beatles, and *I Turn My Camera On* by Spoon. These span music from hard rock to pop and years from the 60s (*Come Together*) up to a few years ago (*I Turn My Camera On*). For each of the four clips, four effects were presented: compression/expansion of spectra and/or gain curves, scaling of NMF components based on their spectral centroid, scaling the components based on the kurtosis of their spectra or gains, and the proposed distortion effect. Therefore the stimuli consisted of a total of twenty clips including the four original versions and all of the effects.

With the compression/expansion there was obviously a choice between compression and expansion, but there was additional variability in that either could be done to the spectra, the gain curves, or both. These different combinations resulted in drastically different effects. As it would be infeasible to have a sample for each possible combination, suitable parameter combinations were chosen subjectively to exemplify the possibilities of each effect type.

3.2. Subjects

There were ten subjects in total, eight male and two female, aged between 22 and 41. Seven of the subjects were musicians and three

Table 1: Overall results (%) of whether the subjects found the effects interesting and would use them if they were available.

	Interesting	Would use
Distortion	80	64
Comp/Exp	80	59
Spec Cent	55	24
Kurtosis	73	39

Table 2: Overall results (%) of the subjects ranking the effects from the most to the least interesting.

	Most	2 nd Most	3 rd Most	Least
Distortion	45	22.5	12.5	20
Comp./Exp.	22.5	32.5	27.5	17.5
Kurtosis	27.5	27.5	25	20
Spec Cent	10	12.5	35	42.5

were not. Seven out of the ten said they were familiar with the term “audio effect” and how they are used, and six of them said they had experience using them.

3.3. Experimental Setup

The listening test was conducted completely on-line. The order of presentation was randomized for each clip. The first question asked simply whether the listener found the effect “interesting.” The next question asked the listeners to describe in their own words the differences they heard between the original and the affected clips. The third question asked the listener whether they would be interested in using the effect were it available as a commercial product. After all of the effects were evaluated for each clip, the subject was asked to rank the four effects for that clip from the most to the least interesting.

3.4. Results

Table 1 shows the percentages of subjects that a) found the effects interesting and b) would consider purchasing or using the effect if it were available to them. For the latter, responses were excluded from subjects who reported “I don’t regularly use audio effects”. In each case the majority of the subjects found the effect to produce interesting results, albeit a slight majority in the case of the spectral centroid effect. A majority of the subjects who use audio effects would be interested in using the distortion and compression/expansion effects. This was not the case, however, with the spectral centroid and kurtosis effects.

Table 2 shows how people ranked the effects from the most to the least interesting. Again the results have been averaged over all the four clips. It is clear that the distortion effect leans towards being the one considered most interesting and the spectral centroid effect the least, with the compression/expansion and kurtosis effects having a fairly even spread.

When describing the differences the subjects heard between the original and affected versions it was common for the subjects to describe the spectral centroid effect as sounding like a simple filter was applied. This would explain the poor results, as listeners familiar with audio effects might find it trivial. On the other hand,

the subjects generally seemed to be intrigued by the distortion effect, as if it were something they had never encountered before.

The responses for the compression/expansion and kurtosis effects were more mixed but still generally positive, and this was also reflected in the written responses. These effects, similar to the distortion effect, found the subjects coming up with more sophisticated descriptions of the things they heard, even in some cases stating that they could not really describe what was going on. For example with the compression/expansion effect for “Smells Like Teen Spirit” two separate responses were received in which the effect was described as making the clip sound more “industrial”; other responses described the clip as sounding like it was “recorded underwater” and “playing inside a can”.

4. CONCLUSIONS

Non-negative matrix factorization provides a musically meaningful representation for audio signals that has not been fully utilized for audio effects. Three different types of effects were investigated in this paper: compression/expansion of component gains and/or spectra, scaling components based on ordering them according to extracted acoustic features, and distortion inherent to the NMF approximation. The distortion effect produced the best results, with the subjects consistently ranking it as one of the more interesting effects. The results concerning the compression/expansion and kurtosis ordering effects were fairly mixed but generally positive.

Future work involves a real-time implementation of the proposed effects. NMF is inherently a non-causal method since the component spectra and gains are estimated jointly. However, a causal (real-time) implementation can be achieved by keeping the spectral bases \mathbf{B} fixed and updating only the gains \mathbf{G} for each incoming audio frame. The component spectra are then updated only occasionally, for example every 5 seconds based on the preceding 10 second segment. A real-time implementation would be useful for more efficient exploration of the parameter space of the effects.

5. REFERENCES

- [1] U. Zolzer, *DAFX - Digital Audio Effects*, J. Wiley & Sons, West Sussex, UK, 2002.
- [2] D. Lee and H. Seung, “Algorithms for non-negative matrix factorization,” in *Neural Information Processing Systems*, Denver, USA, 2001, pp. 556–562.
- [3] Paris Smaragdis and J. C. Brown, “Non-negative matrix factorization for polyphonic music transcription,” in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, USA, 2003.
- [4] T. Virtanen, *Signal Processing Methods for Music Transcription*, chapter Unsupervised Learning Methods for Source Separation, pp. 267–298, Springer, NY, USA, 2006.
- [5] N. Bertin, R. Badeau, and E. Vincent, “Enforcing harmonicity and smoothness in Bayesian non-negative matrix factorization applied to polyphonic music transcription,” *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 538–549, 2010.
- [6] B. C. J. Moore, Ed., *Hearing—Handbook of Perception and Cognition*, San Diego, California, 2nd edition, 1995.
- [7] U. Zolzer, *Digital Audio Signal Processing*, J. Wiley & Sons, West Sussex, UK, 1997.

BLOCK PROCESSING STRATEGIES FOR COMPUTATIONALLY EFFICIENT DYNAMIC RANGE CONTROLLERS

Germán Ramos,

ITACA Institut, Universitat Politècnica de València

Valencia, Spain

gramosp@eln.upv.es

ABSTRACT

This paper presents several strategies for designing Dynamic Range Controllers when using a block-based processing scheme instead of sample-by-sample processing scheme. The processes of energy measurement, gain calculus, and time constant selection are executed only once per each new incoming block of samples. Then, a simple and continuous gain update is computed and applied sample-by-sample between continuous sample blocks to achieve good sound quality and performance. This approach allows reducing the computational cost needs while maintaining the flexibility and behavior of sample-by-sample processing solutions. Several implementation optimizations are also presented for reducing the computational cost and achieving a flexible and better sounding dynamic curve using configurable soft knees or gain tables. The proposed approach has been tested and implemented in a modern DSP, achieving satisfactory results with a considerable computational costs saving.

1. INTRODUCTION

Dynamic Range Controllers (DRC) are often used in audio applications with the objective of mapping an incoming dynamic range to a different outgoing one. They are used in systems like compressors, expanders, noise-gates, limiters, or even all together in a general DRC [1].

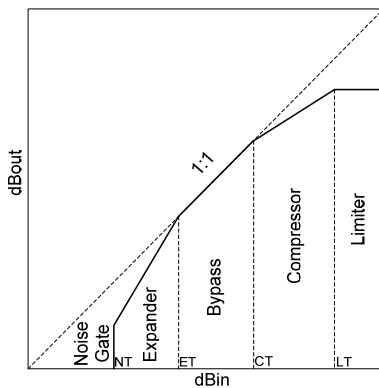


Figure 1: Dynamic Range Controller (DRC) dB input to dB output relationship – Static Curve.

Figure 1 shows the typical dB input to dB output relationship of a DRC that is defined by its Static Curve. It includes all the commented behaviours: noise gate, expander, compressor and limiter. The input levels NT, ET, CT, and LT are the threshold levels in which each behaviour is obtained respectively. The dB

gain applied in each case is obtained as the dB difference from the bypass line (1:1) to the output dB level.

Actually most of the DRC implementations are carried out in the digital domain using Digital Signal Processors (DSP) or microprocessors. Several generic implementations have been proposed [1]-[5]. An interesting improvement that avoids the possibility of clipping the output signal in digital limiters is presented at [6]. The use on non-linearities in DRC with a power polynomial approximation is proposed at [7], [8] with the objective of simulating the non-linear behaviour of analog components like tube amplifiers [2]. [9] proposes the use of a time-varying loudness model in the level detection stage of DRC. Recently, a hardware implementation in a FPGA (Field Programmable Gate Array) of a DRC has been described at [10]. A different approach is [11] that proposes a multichannel DRC working in the frequency domain using frequency warping in order to achieve a closer behaviour to the auditory Bark scale.

This paper presents implementation strategies for DRC when working in block-sample processing schemes instead of classical sample-by-sample schemes. As the energy of the input signal has a considerable lower bandwidth than the signal itself, the processes of energy measurement, gain calculus, and time constant control, are executed only once per each new block of samples, instead of every new input sample. The gain applied is then interpolated between consecutive sample blocks to have a continuous update value and better sounding. By this way a great computational cost saving is obtained while maintaining the desired behaviour of the DRC. Several implementation optimizations are also detailed with the aim of reducing the demanded computational cost, together with a mathematical development of a configurable soft-knee characteristic for the Static Curve.

The paper is organized as follows. Section 2 makes an overview of a sample-by-sample DRC implementation. Section 3 explains the proposed modifications for a block-processing DRC. A mathematical development of a soft-knee DRC is described at Section 4. An implementation in a modern DSP is commented at Section 5. Finally Section 6 summarizes the conclusions.

2. SAMPLE PROCESSING OPERATION SCHEME

Figure 2 displays the scheme of a DRC that is executed for any new input sample $x[n]$ to produce the processed output $y[n]$. An optional post-gain (not shown in the figure) could be applied after $y[n]$ to move up or down the whole Static Curve of Figure 1. Following the implementation of [1], the level of the input $x[n]$ is measured using a RMS detector or a peak detector, giving the input level value $x_l[n]$. This level value is converted to dB and used as the input to the Static Curve to determine the output dB level and hence the needed dB gain that is converted to its linear value $g[n]$. This gain is smoothed to $gs[n]$ with the Smooth Attack/Release block, which controls the dynamic behaviour of the

DRC with the proper selection of the attack and release time constants involved. See implementation details and time constant calculations and recommendation values at [1]. Finally, the smoothed gain $gs[n]$ is applied to the input signal $x[n]$ that could be delayed with the Look-ahead delay block in order to anticipate the behaviour of the DRC and avoiding big transients to pass without being controlled. All of these processes are executed for each new input sample, demanding computational cost.

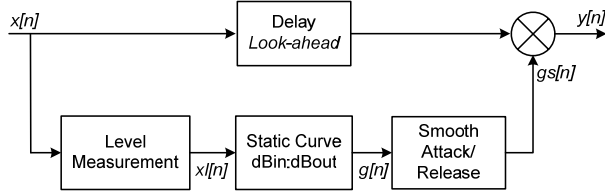


Figure 2: Sample-by-sample DRC operation scheme.

As commented, the RMS or peak level $x[n]$ has a considerable lower bandwidth than the incoming signal itself $x[n]$, and there is no reason in executing the dB conversion, the dB gain calculus and its conversion to linear, and the gain smoothing, for every new sample at the sampling frequency rate of the system fs . To save computational cost, Zölzer proposes at [1] the modified implementation scheme of Figure 3.

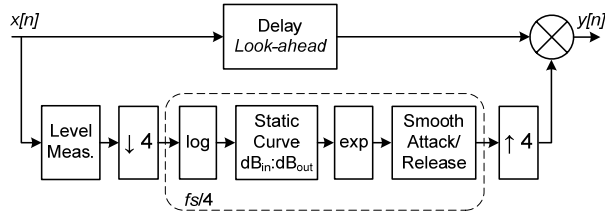


Figure 3: Modified DRC operation scheme with down-sampling and up-sampling for the gain calculus.

Once measured the level of the signal, a decimation by a factor 4 is applied for reducing the internal sampling frequency to $fs/4$ and making all the gain calculus processes. Then, a final interpolation by 4 is executed in order to apply the gain at fs . In this case, all the time constants must be calculated considering its internal $fs/4$ sampling frequency. For each new input sample $x[n]$, the level measurement, the down-sampling and up-sampling, and the gain multiplication are executed at a rate of fs . The four down-sampled processes (dB conversion, dB gain calculus, conversion to linear, smoother) are executed cyclically with a task

scheduler, doing only once of them for each new $x[n]$ cyclically. As a result, the computational cost is reduced because only one of the four processes is executed each time. This paper tries to go one step further increasing the practical decimation ratio as seen at the next section.

3. BLOCK PROCESSING PROPOSED SCHEME

Most of the actual DSP and microprocessors used in audio, due to their internal hardware architecture and the possibility to use software pipeline techniques, are computationally more efficient when working in a frame basis in blocks of N samples, instead of working sample-by-sample. By this way they are also able to use the Direct Memory Access (DMA) engine to move all the audio in and out without the intervention of the CPU. Usual values for N in live audio applications are 16, 32 or 64. The introduced latency in samples is $2 \cdot N$. With $fs=48\text{kHz}$ it is 0.67ms, 1.33ms, and 2.66ms respectively, and with $fs=96\text{kHz}$, 0.33ms, 0.67ms, and 1.33ms. This latency must be increased with the one introduced by the AD and DA converters that is usually below 1 ms. These latency values are considered acceptable for live use.

The proposed block-processing scheme is at Figure 4. The incoming data is the N samples block vector $x[n]$ to $x[n-(N-1)]$. Now, only the optional look-ahead delay and the final gain interpolation and application are executed once per sample at fs rate. The rest of the process will be only executed once per block of N samples, with an effective rate of operation of fs/N .

First, the signal level of the block is measured. If an RMS detector is used, then the x_{RMS}^2 is computed as

$$x_{RMS}^2 = \frac{1}{N} \sum_{i=0}^{N-1} x^2[n-i]. \quad (1)$$

This x_{RMS}^2 is averaged with a first-order low-pass filter to have an energy value with an estimation time longer than N samples. The measured value xl_{block} is obtained with the difference equation (2) where TAV is averaging coefficient [3] and $xl_{block-1}$ is the value of xl_{block} in the previously processed block. For calculating TAV, the effective sampling frequency is now fs/N .

$$xl_{block} = (1 - \text{TAV}) \cdot xl_{block-1} + \text{TAV} \cdot x_{RMS}^2 \quad (2)$$

If a peak detector is used (i.e. in a limiter case), then the maximum absolute value of the input vector is obtained and the peak detector proposed at [1] and [3] is used.

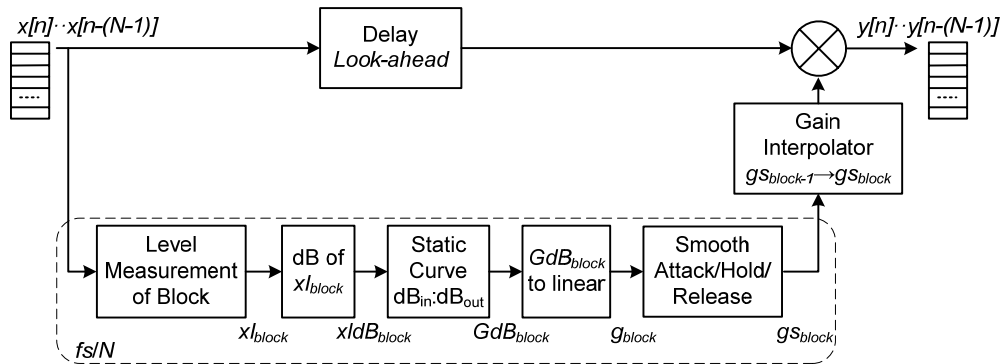


Figure 4: Proposed block-sample DRC operation scheme.

The measured level xl_{block} is now converted to $xldb_{block}$ in dB, using a fast approximation by series expansion [1] or a log table in memory. In case of RMS detector, xl_{block} is the squared RMS measurement and the obtained $xldb_{block}$ must be multiplied by 0.5 to calculate the square-root and achieve the RMS value.

The dB value of the input signal $xldb_{block}$ is used as input to the Static Curve (i.e. like Fig. 1) to determine the output dB level and hence the dB gain to be applied GdB_{block} as the difference between the output dB value and the input dB value. One possibility to implement the Static Curve is using the simple linear equations of the lines of Fig. 1 in each region of operation (noise-gate, expander, bypass, compressor, limiter) using the threshold and ratio values of each region [1]. A more flexible choice is using a dB table in memory, mapping the incoming dB level directly to the applied dB gain. This allows the creation of any kind of Static Curve like the one seen at Figure 5. In this example, the dB gain table is defined every 3 dB, obtaining the dB gains between the table values by interpolation.

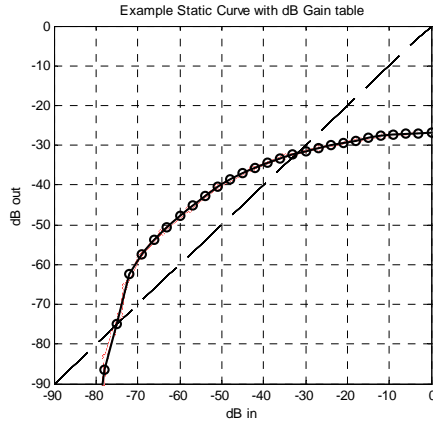


Figure 5: Example of Static Curve implemented with a dB gain table defined with 3dB steps.

The GdB_{block} value is converted to its lineal value g_{block} using again a series expansion or an antilog table. Finally this gain is smoothed to gs_{block} with the Attack/Hold/Release block with equation (3) where $gs_{block-1}$ is the gain of the previously processed block of samples. It controls the dynamic behavior of the DRC with the selection of the time constants AT (Attack Time, signal level increases) and RT (Release Time, signal level decreases).

$$\begin{aligned} gs_{block} &= (1-k) \cdot gs_{block-1} + k \cdot g_{block} \\ g_{step} &= gs_{block} - gs_{block-1} \\ k &= \begin{cases} AT & \text{if } g_{step} < 0 \\ RT & \text{if } g_{step} \geq 0 \end{cases} \end{aligned} \quad (3)$$

A Hold Time HT is included for controlling the time that must stay in the release state before the gain starts to recover, maintaining its gain value. This avoids continuous gain changes and decreases the distortion introduced by the DRC. A good option is using a counter for the HT , being each count value the time of a block of samples of N/fs seconds.

Once calculated the gain gs_{block} for each new input N samples block, it must be applied to the input vector $x[n:n-(N-1)]$ to give the output vector $y[n:n-(N-1)]$. This gain, calculated at a rate of fs/N , is up-sampled (smoothed) to fs with the Gain Interpolator

block. It performs a linear interpolation between $gs_{block-1}$ to gs_{block} just adding to the applied gain the gain step g_{step}/N for each new output sample. A simulation of a limiter with this interpolation is at Figure 6 where the calculated gs_{block} every N samples is displayed (the stepped gain) together with the linearly interpolated smoothed gain. More complex interpolation methods like splines could be used at expense on increasing the computational cost.

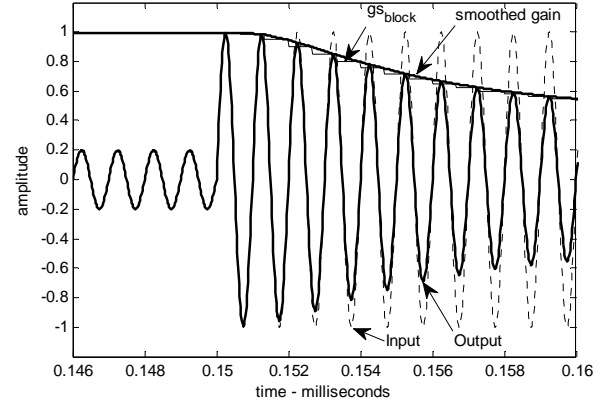


Figure 6: Example of a Limiter with the proposed gain interpolation between consecutive N samples blocks.

With conventional audio content, the proposed block-processing DRC implementation with N values lower than 1ms (i.e. $N=32$ with $fs=48\text{kHz}$ is 0.66ms.), achieves quasi-identical sounding results than a sample-by-sample DRC implementation, with a considerable reduction in computational cost. The effect of the block-sample processing is a slight increase in the effective time constants that are low-limited by the N value. Usually a value greater than 1ms is used for the attack-time. For greater N values, the input vector must be split in smaller blocks and repeat the process once per each smaller block. Other ways the time constants are excessively smeared and it will not be possible to work with short attack times below N/fs that are needed for example in limiters or when working with high frequency signals.

4. SOFT-KNEE STATIC-CURVE DEVELOPMENT

Figure 1 shows a typical Static Curve made of straight lines on each section with sharp transitions (sharp gain changes) between regions (i.e. bypass to compressor). These transitions are referenced as hard-knee. Some DRC use what is called soft-knee transitions that vary progressively from the slope of one region to the slope of the next one. This section describes a procedure to design configurable soft-knee links between consecutive regions of a Static Curve using a parabola as the link function.

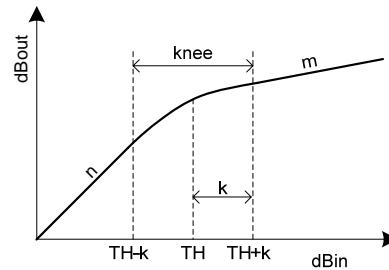


Figure 7: Soft-knee Static Curve design.

Figure 7 shows a soft-knee link of width $knee=2 \cdot k$ between two regions of slopes n and m (the ratio is the inverse of the slope). The threshold dB input is TH , and the soft-knee link is carried out within the input region $TH-k$ to $TH+k$ dB. The parabola (4) is used as the link function and its coefficients are calculated solving the equation system that equals the derivative of the parabola to n at $TH-k$, to m at $TH+k$, and forces the value of the parabola at $TH-k$ or $TH+k$. Once defined the soft-knee Static Curve, the gain in dB is obtained again as the difference between the output and input dB levels. An example of a soft-knee limiter ($n=1$, $m=0$, $TH=-12$ dB) with different k values (from 0 to 10 dB) is displayed at Figure 8. This soft-knee limiter configuration allows arriving to the limit value gradually, not instantly as happens with hard-knee limiters. It works like a compressor that increments continuously its ratio from 1:1 to ∞ :1 in $2 \cdot k$ dB input range, and it is judged to have a better sounding behaviour.

$$dBout = a + b \cdot dBin + c \cdot dBin^2$$

$$a = -1/(4 \cdot k) \cdot (2 \cdot TH \cdot k \cdot (n+m-2) + (TH^2 + k^2) \cdot (n-m)) \quad (4)$$

$$b = 1/(2 \cdot k) \cdot (TH \cdot (n-m) + k \cdot (n+m))$$

$$c = -1/(4 \cdot k) \cdot (n-m)$$

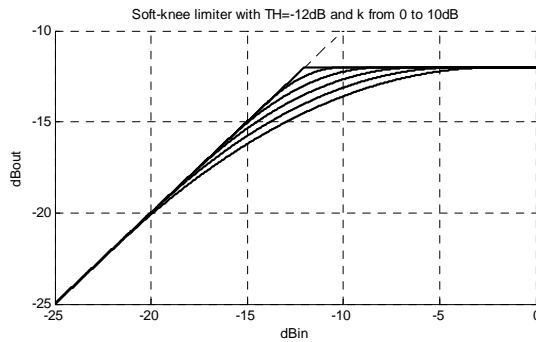


Figure 8: Example of a Limiter with Soft-knee.

5. DSP IMPLEMENTATION

An efficient implementation of the proposed block-based DRC with $N=32$ and $f_s=48$ kHz has been carried out in a SHARC ADSP21489 DSP [12] taking profit of its SIMD (Single Instruction Multiple Data) architecture, and its DMA engine.

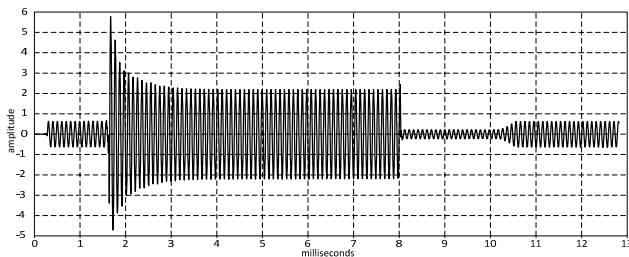


Figure 9: Behavior of the proposed DRC on a DSP with a sinusoidal burst as input

The implementation was able to work with Static Curves defined by lines as Figure 1 with or without soft-knee, and with a gain table defined by the user. The percentage of CPU used by the DRC was of only 0.25%, meanwhile a sample-by-sample version of the DRC took 2.1%. The behavior of the DSP implementation with a sinusoidal burst input signal is shown at Figure

9. It is easy to observe the level control of the DRC and the effect of the attack, release and hold times.

6. CONCLUSION

An efficient implementation of a DRC is proposed in this paper that exploits the benefits of working in blocks of N samples instead of processing sample-by-sample. Most of the processes involved in a DRC are modified and executed only once per new block of samples, and only the final gain smoothing and the application of the gain is executed once per sample. This allows saving computational cost while maintaining the DRC behaviour and sound properties. A generic soft-knee link parabola is also presented. Finally, the proposed DRC has been implemented and tested in an actual DSP verifying the computational cost saving.

7. ACKNOWLEDGEMENTS

This paper has been supported in part by the project PAID-05-10 of the Universitat Politècnica de Valencia, Spain. The author wishes to acknowledge Analog Devices Inc. for the tools and support in the DSP implementation, and to the reviewers for their valuable comments.

8. REFERENCES

- [1] U. Zölzer, *Digital Audio Signal Processing*, John Wiley and Sons, New York, second edition, 2008
- [2] U. Zölzer, *DAFX: Digital Audio Effects*, John Wiley and Sons, New York, 2002
- [3] G. W. McNally, "Dynamic Range Control of Digital Audio Signals," *J. Audio Eng. Soc.*, vol. 32, no. 5, pp. 316–327, May 1984.
- [4] E. F. Stikvoort, "Digital Dynamic Range Compressor for Audio", *J. Audio Eng. Soc.*, vol. 34, no. 1/2, pp. 3–9, Jan./Feb. 1986.
- [5] S. J. Orfanidis, *Introduction to Signal Processing*, Prentice Hall, New York, 1996.
- [6] P. Härmäläinen, "Smoothing of the Control Signal without Clipped Output in Digital Peak Limiters", in *Proc. Digital Audio Effects (DAFx'02)*, Hamburg, Germany, Sep. 2002.
- [7] J. Schimmel, "Using Nonlinear Amplifier Simulation in Dynamic Range Controllers", in *Proc. Digital Audio Effects (DAFx'03)*, London, UK, Sep. 2003.
- [8] J. Schimmel, "Non-linear Dynamics Processing", in *Proc. 114th Convention of the Audio Engineering Society*, 2003
- [9] R. J. Cassidy, "Dynamic range compression of audio signals consistent with recent time-varying loudness models," in *Proc. Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol.4, no., pp. iv-213- iv-216 vol.4, 17-21 May 2004
- [10] A. De Stephanis, M. Conti, M. Caldari, F. and Ripa, "Design refinement for the development of an audio dynamic range controller," in *Proc. Intelligent Solutions in Embedded Systems (WISES), 2010 8th Workshop on*, vol., no., pp.85-90, 8-9 July 2010.
- [11] J. M. Kates, K. H. Arehart, "Multichannel Dynamic-Range Compression Using Digital Frequency Warping", *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 3003-3014, 2005
- [12] <http://www.analog.com/en/embedded-processing-dsp/SHARC/ADSP-21489/processors/product.html>

A PHYSICALLY-MOTIVATED TRIODE MODEL FOR CIRCUIT SIMULATIONS

Kristjan Dempwolf and Udo Zölzer

Dept. of Signal Processing and Communications,
Helmut Schmidt University Hamburg
Hamburg, Germany
kristjan.dempwolfludo.zoelzer@hsuhh.de

ABSTRACT

A new model for triodes of type 12AX7 is presented, featuring simple and continuously differentiable equations. The description is physically-motivated and enables a good replication of the grid current. Free parameters in the equations are fitted to reference data originated from measurements of practical triodes. It is shown, that the equations are able to characterize the properties of real tubes in good accordance. Results of the model itself and when embedded in an amplifier simulation are presented and align well.

1. INTRODUCTION

The theory of vacuum tubes was already discussed at full length many decades ago and the best books on these devices were still written in the 1930s to 1950s [1, 2, 3, 4]. The idea of modeling tubes intends not to challenge the correctness of the traditional formulations that can be found in those books. It is more the task of emulating the behavior of tubes at their boundaries of purpose. While the first tube models were inspired by the desire for SPICE-based simulations of hi-fi circuits, the newer approaches are rather motivated by real-time simulations and guitar amplifier circuits. These amplifier designs gain their special sound by intentionally provoking a distortion of the instruments' signal and thus have just a little in common with ordinary amplifier theory. The recent activities towards tube modeling consider the operation in overdriven amplifiers [5].

With the digital simulation of analog audio circuits in mind, the investigation of tube models is a natural next step. Any improvement in the critical parts promises better performance of the complete simulation. Such improvements include challenges such as accuracy, computational complexity and robustness.

The paper is organized as follows: The second section will review the physical fundamentals of vacuum tubes. This is necessary, because for a successful modeling of tubes or tube circuits, a good knowledge of the theory is required. The tube experts may skip this part. In Section 3 practical tubes are discussed. After this some modeling approaches are reviewed in Section 4 and the new contribution is introduced in Section 5. Finally, in Section 6, some results are presented, utilizing the model equations in the state-space model of a common amplifier stage.

2. VACUUM TUBE BASICS

In this section we give a very brief introduction to the fundamentals of vacuum tubes. For detailed exploration we refer to the historic standard literature [1, 2, 3, 4]. Because of a certain contri-

bution to our triode model, we start with a view to simple vacuum diodes.

2.1. Diode Characteristics

The most simple vacuum tube, the so-called vacuum diode, consists of only two electrodes which are mounted in a vacuum cylinder. One electrode, the *cathode* (K), is heated and hence electrons are emitted. The second electrode is called the *anode* or *plate* (A) and is designed to collect the emitted electrons. Since electrons can not be emitted by the cold anode the assumption of unilateral conductivity is valid. In unheated condition, no current flow is possible.

2.1.1. Initial Velocity Current

Without an external voltage applied to the anode, and even for a slightly negative anode with respect to the cathode, some of the faster electrons reach the anode and a small current flow is observed. A small negative voltage must be applied in order to suppress the current flow. The current through the diode I follows the exponential equation

$$I = I_0 \cdot e^{\frac{V}{E_T}}, \quad (1)$$

with anode voltage V , thermal voltage E_T (in Volt) and current I_0 at zero voltage.

2.1.2. Space-Charge Current

When a positive voltage is applied from anode to cathode, the emitted electrons are pulled by the anode and an increasing current flow is observed. But only a part of the emitted electrons reach the anode. Most of them have only low velocity, they stay near the hot cathode and form the *space charge*, a cloud of negative charges. The resulting current depends on the anode voltage and is self-limited by reason of the electric field of the (negative) charges. This is expressed by the Langmuir-Child equation

$$I = G \cdot V^{\frac{3}{2}}, \quad V > 0, \quad (2)$$

with perveance G , a constant that only depends on the geometrical construction of the tube.

2.1.3. Saturation Current

Beyond a certain voltage, all emitted electrons are drawn to the anode. Saturation appears, that means, the current remains nearly constant when the external voltage is increased further.

2.2. Triode Characteristics

Triodes have a third electrode placed between anode and cathode, which is constructed as a wire mesh and therefor called *grid* (G). A voltage applied to the grid is able to control the flow of electrons from the cathode: a negative grid voltage causes an electric field that counteracts with the electric field from the anode, the cathode current is reduced. A higher negative grid voltage inhibits the current flow thoroughly. Small changes in grid voltage cause high changes in current flow, thus it is possible to use triodes for amplification.

For clarity we define the following conventions: V_a is the anode- and V_g the grid voltage (both referred to the cathode potential), I_k , I_a and I_g are the cathode, anode and grid currents, respectively. We define $I_g + I_a = I_k$ where I_a and I_g are directed into the device.

For the introduction of a grid between anode and cathode it is a common prospect to replace the three-terminal triode by a two-terminal one [2], where the electrode is placed at the grids' position and loaded with the *effective voltage*,

$$V_{\text{eff}} = \left(V_g + \frac{1}{\mu} V_a \right). \quad (3)$$

The amplification factor μ states about how much higher the anode current is influenced by the grid than by the anode voltage. The current I_k is calculated analog to equation (2),

$$I_k = G \cdot (V_{\text{eff}})^{\frac{3}{2}}, \quad V_{\text{eff}} > 0. \quad (4)$$

Equation (4) is valid for the “normal” operating condition, where the grid is slightly more negative than the cathode, while the anode potential is very high (e.g. $V_g = -2$ V and $V_a = 300$ V). As for the diode we speak of the *space-charge region*. The grid draws no current ($I_g = 0$) and consequentially anode and cathode current are equal, $I_a = I_k$.

As a special case we consider the operation with a positive grid. Since the grid now likewise attracts electrons, a positive current I_g arises and hence the cathode current divides into anode and grid current. Most books do not respect this case in detail, because the flow of grid current introduces distortions and thus do not satisfy the classical amplifier theory anymore. The positive grid is irrelevant for the design of linear hi-fi tube amplifiers. Fact is, that most guitar amplifier designs since the 1960s are operated under these conditions and that the grid current is responsible for some distortion effects like the *blocking distortion* [6, 7]. For a correct simulation the inclusion of the grid current can play an important role, especially if coupled stages in a cascade are examined.

3. PRACTICAL TUBES

Real tubes depart occasionally clearly from the idealized formulations given in Section 2 as well as from the information found in the manufacturer's data sheets. Aging effects, small constructive variations and origin have influence on the characteristics, to name a few reasons. Even tubes from the same type and same manufacturer may show up to 20 % deviation of each other. Exemplary measurement results are discussed e.g. in [8, 9].

3.1. Measurements

To have reliable reference data, numerous measurements on standard triodes of type 12AX7 were performed by the authors. Grid

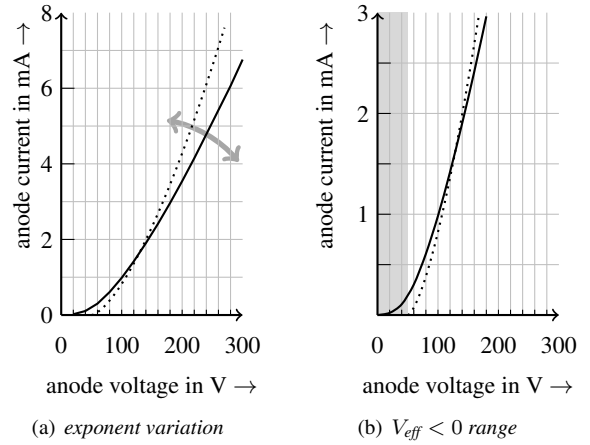


Figure 1: Anode current calculated with Langmuir-Childs formula (dotted) and measurements from a RSD 12AX7 triode (solid) in a qualitative plot.

and anode current were measured at the same time in good resolution for a fine mesh of discrete V_g and V_a voltages. Since tube circuits are normally operated with AC, the measurements were not performed under static conditions, but using switched voltage supplies and triggered meters. The two DC supplies for V_a and V_g were switched on at the same time and the currents were measured for a short time period. From the stored current values the median was chosen to be the final value. The distance between the discrete measurement points was chosen small enough, that a subsequent interpolation is not required. In consideration of the constraints given by maximum power consumption and maximum anode current, the observed working range was $V_a = 20$ V to 300 V and $V_g = -5$ V to 3 V. The discrete steps for the grid were $\Delta V_g = 0.1$ V, $|V_g| < 1$ V and less dense for higher values, and $\Delta V_a = 20$ V for the anode voltage. This manually performed procedure is highly time-consuming (ca. 8h net./system), so only three triode systems were tested as a start.

At first we settle for evaluating the measurements only qualitatively, some complete datasets will be shown in Section 5.5.

3.2. Observations on the Anode Current

Figure 1 opposes qualitatively a measurement from an old 12AX7 tube (RSD) and Langmuir-Child's law. While the measurement in principle follows the formula, two differences are visible: First, the theoretic $\frac{3}{2}$ -exponent does not fit perfectly for real triodes, see 1(a). In the plotted example, the slope of the measured curve is lower, representing an exponent $< \frac{3}{2}$.

The second difference can be found at the bottom of the curve. Equation (4) yields $I_k = 0$ for $V_{\text{eff}} = 0$, and is not defined for voltages $V_{\text{eff}} < 0$. The measurement offers a small current for this case and a smooth transition towards zero current. This is depicted in Figure 1(b), where the highlighted region tags the range $V_{\text{eff}} < 0$. We will resume these points later on in Section 5.

3.3. Observations on the Grid Current

The measured grid current increases when moving from small negative voltages towards the ordinate following an exponential shape,

see Figure 2(a). For positive voltages the current increases less steep, but may reach significant values (Figure 2(b)). The grid current I_g is highly influenced by V_g , but has only small dependency on V_a .

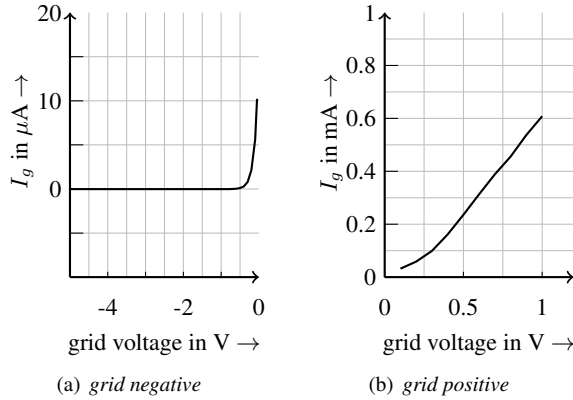


Figure 2: Grid current measured for 12AX7 triode (same tube).

4. MODELING OF TUBES

In the last 20 years about a dozen different tube models for SPICE simulation tools were invented. These mathematical expressions are also widely used in digital audio effect algorithms. For application of those SPICE models we refer to a recent book [9] and the original papers.

Basically we can distinguish between *physical* models, which are based on the physical tube equations and *heuristic* or *phenomenological* models, which have no physical foundation. A different distinction can be made by classifying models by purpose of use. While the first tube models were motivated by SPICE-based hi-fi circuit analysis, the newer models are explicitly invented for guitar amplifier simulation. In the following we will review three approaches briefly to get an introduction to the subject.

One of the most popular models is a phenomenological description invented by Koren [10]. In his approach the triode is composed from basic SPICE components, namely current sources, resistors and diodes and a mathematical description. The work features a library with popular tubes. Koren's model gives good results for simulations with negative grid.

A recent model was proposed by Cardarilli *et al.* [11]. In their formulation the “tube constants” (e.g. μ , G) are exchanged by 3rd-order polynomials, then the model (with its many unknowns) is fitted to measurement data. The presented results, including the simulation of a guitar amplifier, are very promising. Nevertheless the formula has lack of physical interpretation, because the polynomial for the perveance, which is the only real constant, has a high excursion.

Cohen and Helie [12] extended Koren's model by a more realistic grid current, using a piecewise-defined function with a linear part, a second-order polynomial and a smooth transition. They performed current measurements in a similar manner, followed by a bilinear interpolation. Based on these data a fitting is executed, identifying Koren's parameters individually for three triodes.

5. NEW TRIODE MODEL

Based on the observations from Section 3 a new triode description is deployed. We follow the idea, that the model in general has to be physically motivated, i.e. the formulations have to follow the traditional equations as explained in the second section. Furthermore the model has to be adaptable, so that simulations can be fitted individually to a selected tube. Last but not least, formulations with low complexity are desirable, to enable real-time applications.

5.1. Cathode Current

The exponent in equation (4) may differ from the theoretical value $\frac{3}{2}$, as already stated in Section 3.2. This can be explained by the fact, that the exponent is calculated for simplified and ideally constructed triodes and thus may differ from practical ones.

Kniekamp focused in a historic study on the exponent and specified several counteracting reasons for the deviation [13], saying that the exponent may be both smaller or greater than 1.5. Reich [1] identified the exponent generously to be approximately in the range 1.2 to 2.5.

In fact, this was already part of other triode models, Koren for example assumed a fixed exponent of 1.4 in his model [10]. In the upcoming model the exponent will be parametrized with γ . This leads to the first approximation for the cathode current

$$I_k \approx G \cdot (V_{\text{eff}})^\gamma, \quad V_{\text{eff}} > 0. \quad (5)$$

Note that the transition for $V_{\text{eff}} \leq 0$ is not considered in this equation.

5.2. Grid Current

With a view to the mechanical construction it is obvious that the relation between grid and cathode can be modeled roughly as a vacuum diode. The variables may differ considerably since the grid electrodes' construction is distinct from the solid anode. But in general any N-electrode arrangement will show an initial velocity current and space-charge effects, cf. [2].

From the measurements we found, that the grid current I_g is highly influenced by the grid voltage, but has only small dependency on the anode voltage. This leads to a simplified approximation for $I_g = f(V_g)$,

$$I_g \approx G_g \cdot V_g^\xi \quad (6)$$

with the grid perveance G_g and exponent ξ . Similar observations are specified in [12] and supported by the literature [2, 4]. Note that equation (6) corresponds to the space-charge law, equation (4), but with possibly deviant exponent. Near to $V_g \approx -0$ the grid current was measured to resemble an exponential curve. This can be explained by the initial velocity current, as in equation (1).

5.3. Smooth Transition

To create a smooth transition between piecewise functions various approaches are possible. For our model we decided to use a combination of exponential function and logarithm. As introduction we examine a simple function $f(x)$ with

$$f(x) = \begin{cases} x & , x > 0 \\ 0 & , x < 0. \end{cases} \quad (7)$$

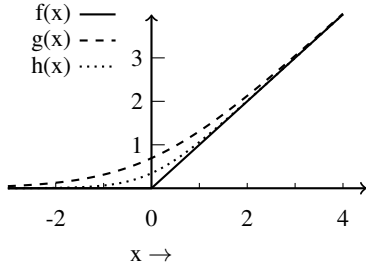


Figure 3: Smoothing function.

Obviously there is a knee at $x = 0$. To smoothen this discontinuity we consider a second function

$$g(x) = \log(1 + e^x) \quad (8)$$

showing the same tendencies as equation (7) for greater values of $|x|$. With increasing x the result tends to $g(x) \approx \log(e^x) = x$, while the exponential and thus the function value approaches zero in $-x$ direction.

The curve shape can furthermore be adapted towards the linear function using an additional factor C , leading to

$$h(x) = \log(1 + e^{C \cdot x}) \cdot \frac{1}{C}. \quad (9)$$

This relation is illustrated in Figure 3, where equations (7), (8) and (9) are plotted (adaption factor was chosen $C = 2$).

With this mathematical trick the smoothing of our modeling equations comes to mind. The idea is not new, by taking a closer look to Koren's model, we figure out equation (8) in the expression for the anode current. Evidently equation (9) with the extension is more flexible.

5.4. Final Equations

The derivations for the cathode and the grid current are now assembled and the smoothing is applied to our formulations for I_k and I_g . Enhancing equation (5) and equation (6) yield the final formulations:

$$I_k = G \cdot \left(\log \left(1 + \exp \left(C \cdot \left(\frac{1}{\mu} \cdot V_a + V_g \right) \right) \right) \cdot \frac{1}{C} \right)^\gamma \quad (10)$$

$$I_g = G_g \cdot \left(\log \left(1 + \exp \left(C_g \cdot V_g \right) \right) \cdot \frac{1}{C_g} \right)^\xi + I_{g0}, \quad (11)$$

with the perveances G and G_g , exponents γ and ξ and the adaption factors C and C_g . For the grid current a constant I_{g0} is added due to stability reasons. The anode current subsequently has to be the difference of both, giving

$$I_a = I_k - I_g. \quad (12)$$

These equations still reveal the well-known physics but with some degree of freedom. Furthermore, they are continuously differentiable and have no discontinuities.

5.5. Fitting to the Measurements

The model equations feature four free parameters for the cathode current (G, C, μ, γ) and four for the grid current (G_g, C_g, ξ, I_{g0}).

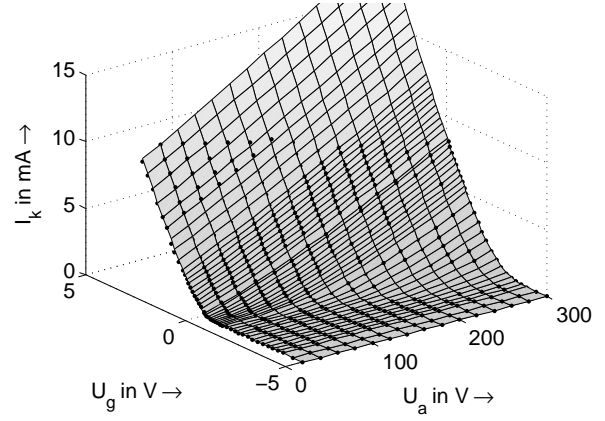


Figure 4: Fitting of equation (10) to measured I_k .

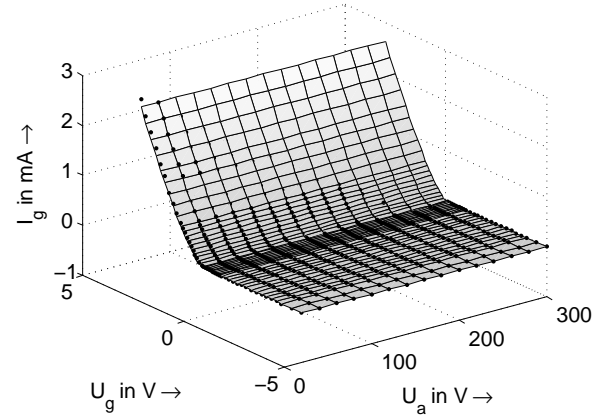


Figure 5: Fitting equation (11) to measured I_g .

Using a curve fitting algorithm like `sftool`¹ it is now possible to adapt the parameters to the measurement data. Figure 4 shows both measurement data and fitting curve of the cathode current for a 12AX7 triode, displayed as a 3-D plot. The surface represents the analytically computed characteristics according to equation (10) while the measurements are given as black dots. The influence of the grid voltage on the current is obvious. Figure 5 displays the same for the grid current, revealing that I_g is almost independent of V_a , as mentioned before. It can be asserted that the measurement dots align well with the surfaces for both plots.

The fitting results for three systems are given in Table 1. Fortunately, G, μ and γ reflect the standard values that can be found in the books in a satisfying approximation.

The 3-D plots are nice for getting an idea of the dependencies, although parametric curves allow for a better inspection of the results. In Figure 6 the complete characteristics of a measured triode are compared to the results from the fitting. The influence of grid and anode voltage on the currents is clearly visible and accurately replicated by the model.

¹Surface Fitting Toolbox for MATLAB, The Mathworks

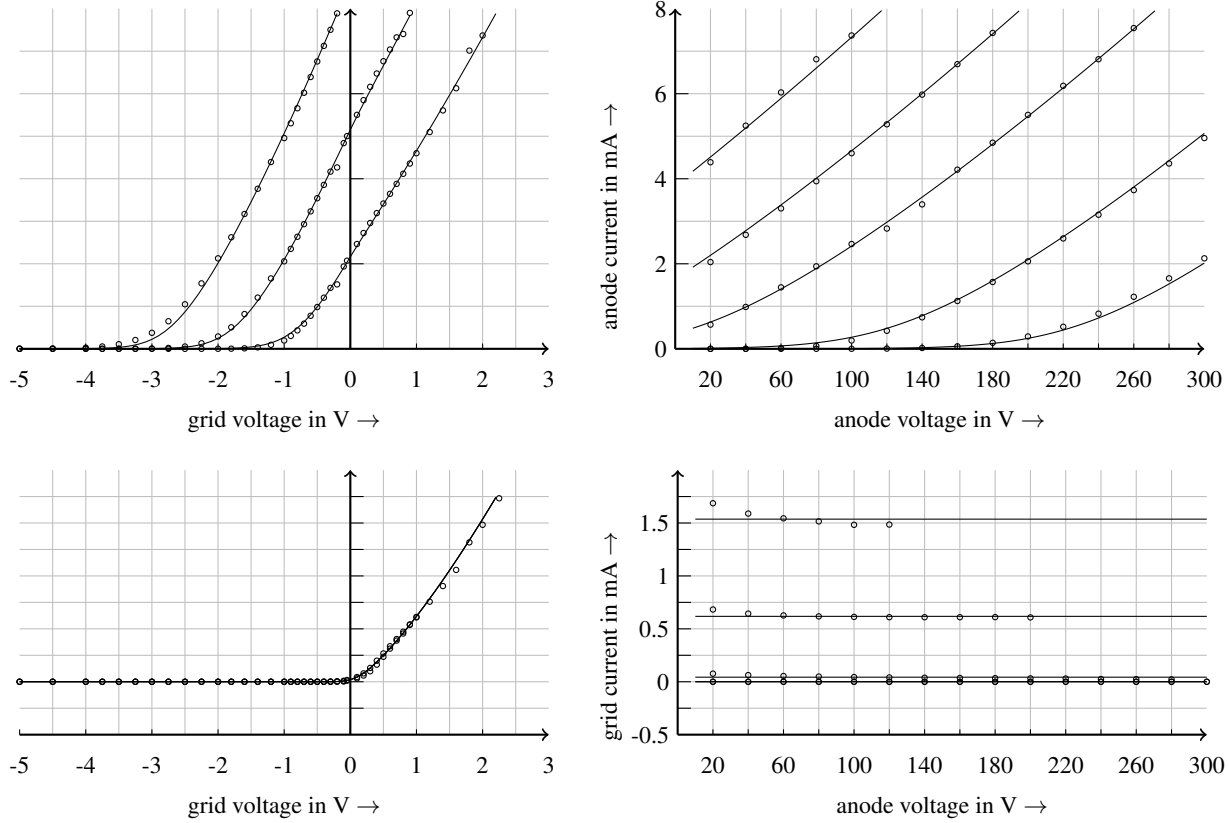


Figure 6: Complete characteristics of a 12AX7 triode. In black: simulation results, \circ : measurement data from a RSD tube. Plots on the left: grid family for $V_a = 100$ V, 200 V and 300 V. Plots on the right: plate family for $V_g = -2$ V, -1 V, 0.1 V, 1 V and 2 V.

	RSD-1	RSD-2	EHX-1
G	2.242E-3	2.173E-3	1.371E-3
μ	103.2	100.2	86.9
γ	1.26	1.28	1.349
C	3.40	3.19	4.56
G_g	6.177E-4	5.911E-4	3.263E-4
ξ	1.314	1.358	1.156
C_g	9.901	11.76	11.99
I_{g0}	8.025E-8	4.527E-8	3.917 E-8

Table 1: Individually fitted parameters for 12AX7 triodes.

5.6. Parasitic Capacitances

The electrodes and their mechanical assembly lead to parasitic capacitances which have to be taken into account for a correct dynamic behavior. As in other existing tube models we assume the standard values that can be found in the data sheets, i.e. $C_{ak} = 0.9$ pF, $C_{gk} = 2.3$ pF and $C_{ag} = 2.4$ pF for 12AX7.

6. RESULTS

6.1. Comparison with other Models

The proposed model has to be compared to the existing approaches. Figure 7 shows again the family characteristics of the new model

(black curves) and the measurements of one tube (circles). The dotted curves show the characteristics of Koren's model [10]. The curves have generally the same progression. Differences are found for the grid current (lower plots) and for the shape of the anode current for positive grid voltages.

This observation is not surprising. The new model was individually fitted to the measurement data, so the black curves should align better with the circles than the general Koren model. More meaningful is the comparison to other individual models. We decided to examine the individual model proposed by Cohen and Helie [12]. It is based on Koren's formula, but with a different grid current where a piecewise-defined function with three subdomains is suggested. To achieve an equitable comparison the equations were fitted in a similar manner to the same measurement data. The results are displayed in Figure 7 as dashed curves. The improvement due to the individualization is clearly visible and the curves align better with the measurements.

Regarding our approach a deviation remains for high anode voltages and negative grid voltage (see upper left plot), where the standard Koren model interestingly performs best. Besides, the shape for very low anode voltages ($V_a < 20$ V) is not captured correctly.

However, in this comparison the proposed equations allowed better fitting results and achieved a good alignment with the measurements. The same observation was made for the other tested triodes.

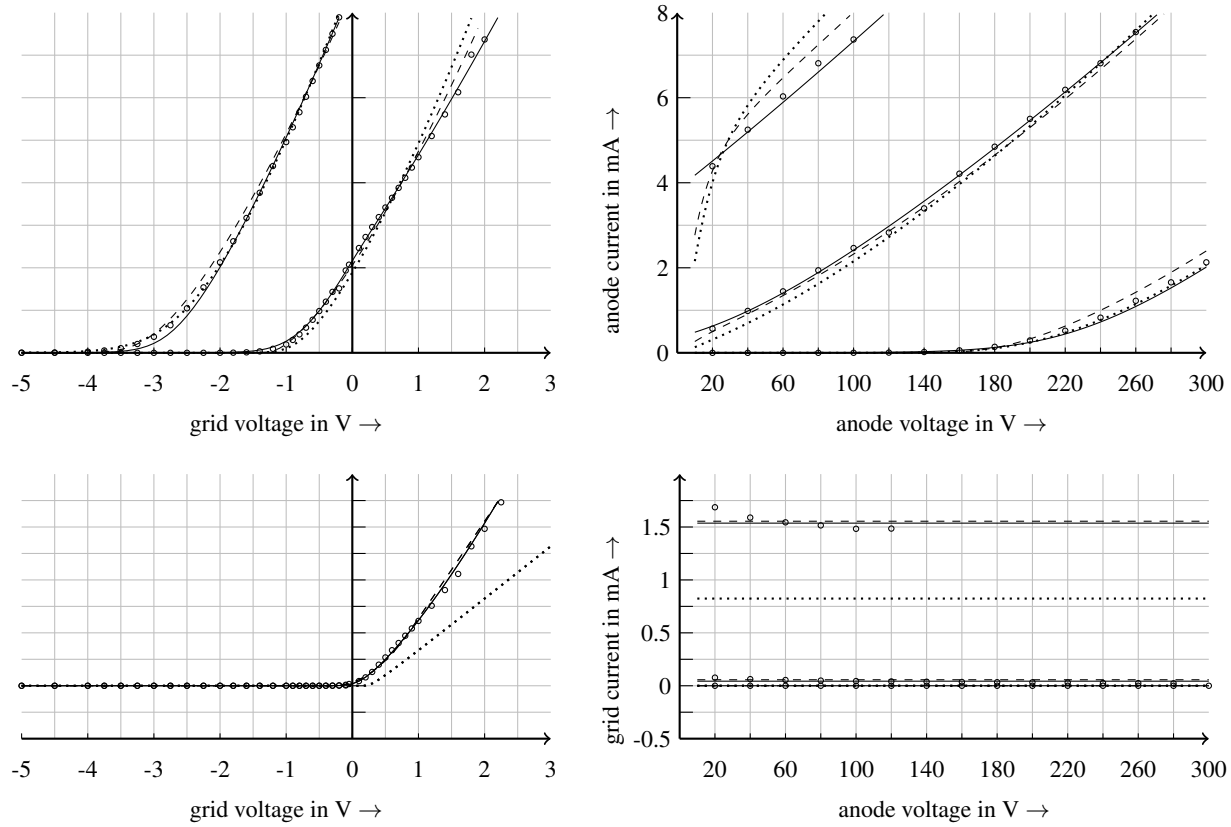


Figure 7: Comparison of different triode models. Plots on the left: grid family for $V_a = 100$ V and 300 V. Plots on the right: plate family characteristics for $V_a = -2$ V, 0.1 V and 2 V. Shown are Koren's approach (dotted), Cohen and Helie (dashed) and the new model (black). The discrete measurement points are marked with \circ .

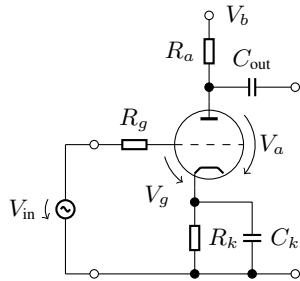


Figure 8: Common-cathode amplifier stage.

6.2. Application: Common-Cathode Amplifier Stage

To check to what extent the model equations replicate real triodes when used within a circuit simulation, a common tube amplifier stage was prototyped. The same tubes as those used in the fitting procedure were operated in this test circuit and the responses to various excitation signals were measured.

Figure 8 depicts the schematic of the chosen amplifier stage. This common-cathode amplifier can be found in almost all preamplifiers designs. We skip a detailed circuit analysis and are content with the information, that this represents an inverting amplifier giving a high gain. The operating point is a bit negative, but for higher

input amplitudes the grid will be temporarily positive. Several papers analyzed this simple but representative circuit and discussed qualified simulation techniques [9, 12, 14, 15].

For the simulation of the circuit a state-space model was implemented. The state-space representation has turned out to be a practical tool for the simulation of nonlinear circuits [16]. We skip the details of the implementation and refer to a recent study dealing with this method [17]. The state-space representation gets along with four state variables, two for the shown capacitors C_{out} and C_k and two for the parasitic capacitances of the triode. In fact, in Section 5.6 we introduced three capacitances (C_{ak} , C_{gk} and C_{ag}) but they are not independent and thus can be reduced to two.

6.2.1. Waveforms

A comparison of time signals is given in Figure 9, where the similarity for different frequencies and input amplitudes is checked. As expected, the waveforms for higher input amplitudes are distorted. The measured and simulated outputs align well and only small differences can be observed. All measurements and simulations were performed with a sampling frequency $f_s = 96$ kHz.

6.2.2. Harmonic Spectra

Waveform comparisons give only limited information on how good a simulation performs. In addition, the harmonic content of the

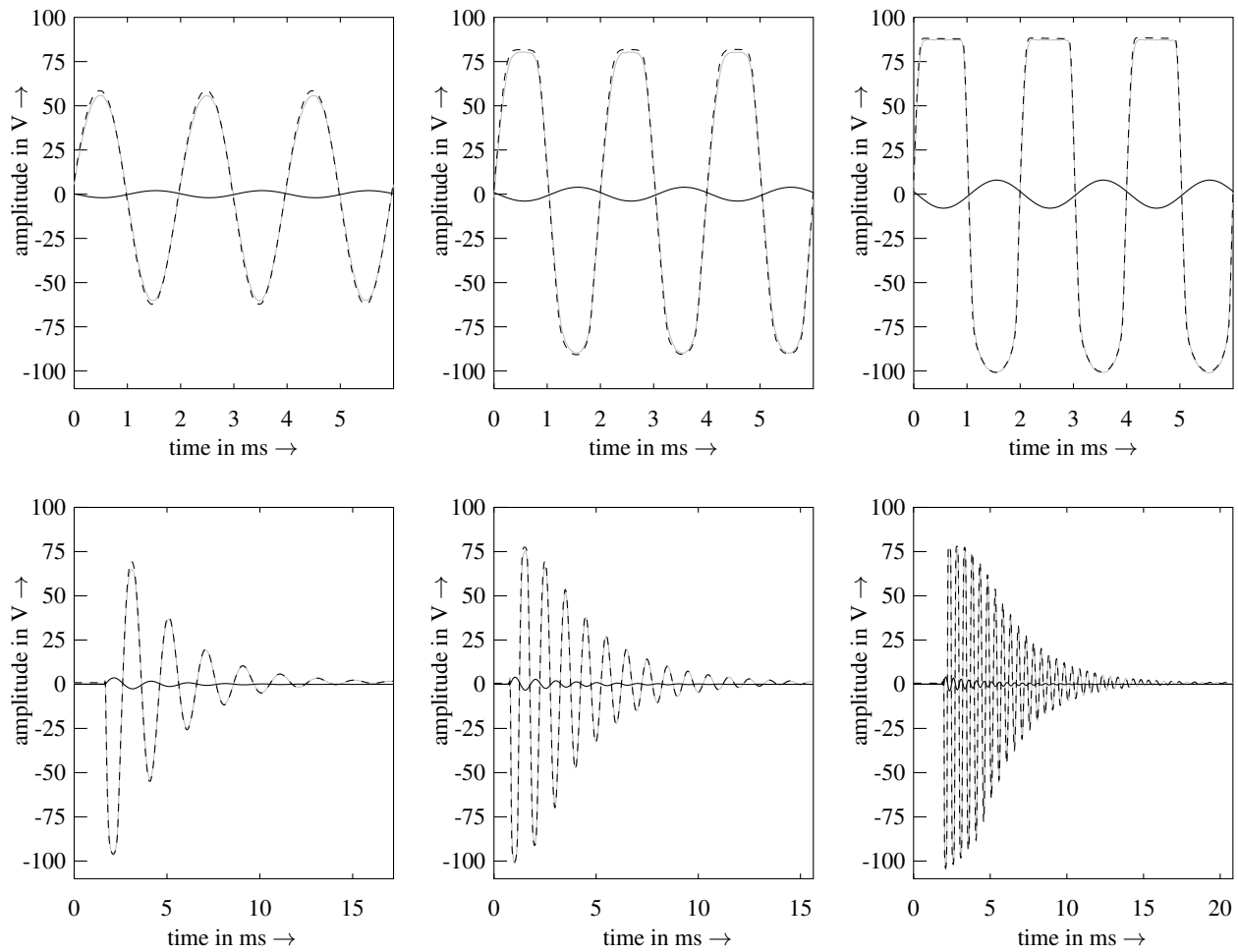


Figure 9: Waveform comparison of measurement (gray) and simulation (dashed black) for different input signals (solid black). First row: sinusoidal excitation with 500 Hz and 2 V, 4 V and 8 V. Second row: 4 V sine burst for frequencies 500 Hz, 1 kHz and 2 kHz.

output is computed both for reference system and simulation. The exponential sweep technique [18] was used to measure the introduced distortion. Figure 10(a) shows the harmonic distortions of the discussed circuit for a 4 V measurement signal. The results of the simulation are given in 10(b). As the plots illustrate, a satisfying similarity is achieved. Differences are visible at high frequencies, where the simulation is a bit flatter.

6.2.3. Guitar Sound

Some sound clips of electric guitar playing are recorded with the circuit and simulated. The comparison supports the good conformance of the results. The clips are available on our homepage <http://ant.hsu-hh.de/dafx2011/tubemodel>

6.3. Limitations

The measured type of triode 12AX7 is known as a *linear* tube [9]. Keeping guitar amplifier distortion in mind this may confuse at first. But the classification linear or nonlinear regime depends on the course of the amplification factor μ . A typical example for a

nonlinear triode is the 12AT7, which can be found in audio circuits (e.g. phase inverter stages) as well. In addition to the 12AX7 measurements, one 12AT7 triode was checked. It was found that the fitting results are not as good as for the 12AX7 tubes. Especially the assumption that I_g is almost independent of V_a is not valid anymore. As a second limitation we have to respect a lower bound for the anode voltage. When operating the triode model in the region with $V_g > 0$ and small anode voltages, say $V_a < 20$ V, the results are obviously not correct. For real triodes the anode current will decrease rapidly when approaching $V_a = +0$ V, an effect that is not reproduced by the presented formulations.

7. DISCUSSION AND OUTLOOK

The proposed equations are able to characterize the behavior of real triodes in a good accordance. One advantage over other descriptions is that the equations can be deduced from the fundamental laws to a large extent.

Though there are some restrictions as identified by the comparisons. As already stated, the shape for very low anode voltages ($V_a < 20$ V) is not captured correctly. But normally the 12AX7

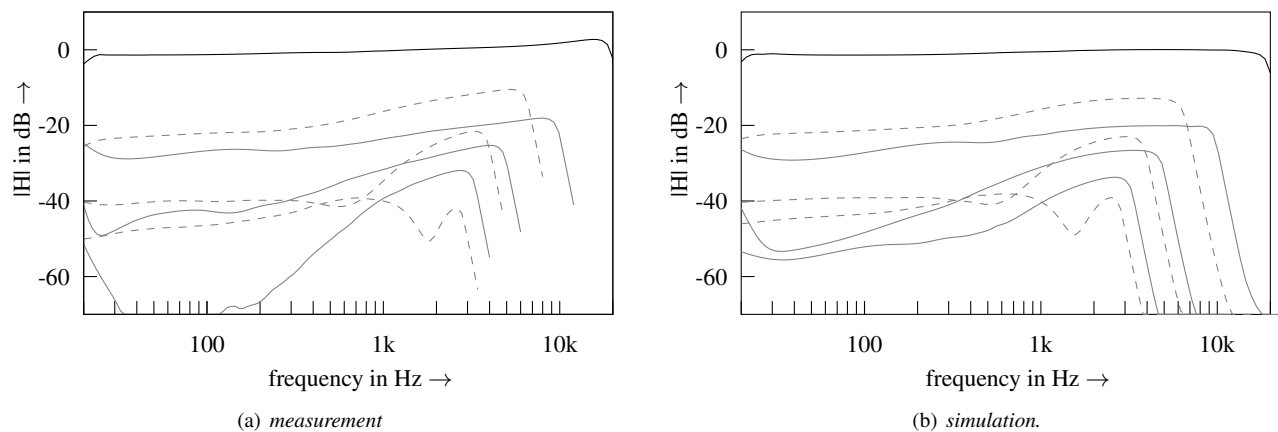


Figure 10: Frequency domain analysis of the common-cathode amplifier. Smoothed harmonic spectra for a sweep with 4 V amplitude. Shown are the fundamental (black), odd (dashed gray) and even order harmonic responses (solid gray).

triode will not be operated in this region, because the internal resistance approaches a finite value for all anode voltages. Hence this shortcoming is considered to be marginal.

As a future prospect more tubes have to be inspected with even higher resolution of the discrete measurement points to improve the quality of the fittings. Furthermore the model has to be enhanced so that nonlinear triodes are comprised, too.

8. CONCLUSION

A new model for triodes of type 12AX7 was presented, featuring a good replication of the grid current and physically-motivated formulations. The equations are mainly based on the Langmuir-Child's law and can be calculated separately for cathode and grid current.

Free parameters within the formulations were used to perform an individual fitting to measurement data of practical triodes. It was shown, that the equations are able to characterize the properties of real tubes in good accordance.

To prove the suitability for the simulation of audio circuits, the model was embedded in a state-space description of a typical tube preamplifier. The simulation results for different input amplitudes and frequencies were compared to reference measurements and showed a good match. The derived equations are simple, continuously differentiable and applicable for real-time simulations.

9. REFERENCES

- [1] H. Reich, *Principles of Electron Tubes*, McGraw Hill, 1st edition, 1941.
- [2] H. Barkhausen, *Lehrbuch der Elektronenröhren und ihrer technischen Anwendungen*, Verlag S. Hirzel, Leipzig, 1945.
- [3] K. Spangenberg, *Vacuum Tubes*, McGraw Hill, 1942.
- [4] H. Rothe and W. Kleen, *Grundlagen und Kennlinien der Elektronenröhren*, Geest & Portig, Leipzig, 1951.
- [5] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music Journal*, vol. 33, no. 2, pp. 85–100, Summer 2009.
- [6] M. Blencowe, *Designing Tube Preamps for Guitar and Bass*, Blencowe, 2009.
- [7] R. Aiken, "What is blocking distortion?," [online], www.aikenamps.com/BlockingDistortion.html, 1999.
- [8] M. Zollner, *Physik der Elektrogitarre*, (preprint), Regensburg, 2010.
- [9] A. Potchinkov, *Simulation von Röhrenverstärkern mit SPICE*, Vieweg + Teubner, 1st edition, 2009.
- [10] N. Koren, "Improved vt models for spice simulations," *Glass Audio*, vol. 5, pp. 18–27, 1996.
- [11] G. C. Cardarilli, M. Re, and L. Di Carlo, "Improved large-signal model for vacuum triodes," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2009, pp. 3006–3009.
- [12] I. Cohen and T. Helie, "Measures and parameter estimation of triodes, for the real-time simulation of a multi-stage guitar preamplifier," in *Proc. 129th AES Convention*, San Francisco, USA, Nov 4-7 2010, number 8219.
- [13] H. Kniekamp, "Die Abweichungen der Verstärker-röhrenkennlinien vom $e^{3/2}$ -Gesetz," *Telegraphen- und Fernsprechtechnik*, vol. 20, no. 3, pp. 71–76, 1931.
- [14] J. Macak and J. Schimmel, "Real-time guitar tube amplifier simulation using an approximation of differential equations," in *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10 2010.
- [15] F. Santagata, A. Sarti, and S. Tubaro, "Non-linear digital implementation of a parametric analog tube ground cathode amplifier," in *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 10-15 2007.
- [16] D.T. Yeh, *Digital Implementation of Musical Distortion Circuits by Analysis and Simulation*, Ph.D. dissertation, Stanford University, June 2009.
- [17] K. Dempwolf, M. Holters, and U. Zölzer, "Discretization of parametric analog circuits for real-time simulations," in *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10 2010.
- [18] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in *108th AES Convention*, Paris, France, Feb. 19-24 2000.

A SIMPLE AND EFFICIENT FADER ESTIMATOR FOR BROADCAST RADIO UNMIXING

Mathieu Ramona*

IRCAM, Centre Pompidou
1, place Igor Stravinsky, 75004 Paris, France
mathieu.ramona@ircam.fr

Gaël Richard

Institut Telecom, Telecom ParisTech
LTCI-CNRS
37-39 rue Dareau, 75014 Paris, France

ABSTRACT

This paper presents a framework for the estimation of the faders gain of a mixing console, in the context of broadcast radio production. The retrieval of the console state is generally only possible through a human-machine interface and does not permit the automatic processing of such information. A simple algorithm is provided to estimate the faders position from the different inputs and the output signal of the console. This method also allows the extraction of an additional unknown input, present in the mix output. An exhaustive study on the optimal parameter setting is then detailed, that shows good results on the estimation.

1. INTRODUCTION

The transition of the broadcast technical craft from analog to digital audio casting is an important imminent change for radio stations in France. Indeed, the forthcoming revolution of the radio media is the emission of associated interactive visual content that provides a live complement to the audio content. The automatic production of additional multimedia content implies an increased control on the whole media production process. Such feature requires the upstream knowledge of the audio media produced and emitted, which is not possible with the actual broadcast model state.

Interfacing with a mixing console is a typical example of this lack. Typically, several inputs of the console are active and dedicated to various audio streams (i.e. jingles, advertisements, liners...) but only a small part of them is actually present in the mix output emitted by the station. This type of material is highly proprietary and an open machine interface is rarely provided to check the state of the controls. However, knowing the exact content of the output is essential to be able to generate data associated.

A typical example of this issue is the displaying of the album covers of a musical playlist, on a multimedia stream coupled with the audio stream. Succeeding musical tracks are assigned to different channels of the console, and mix-faded. The track faders position determine the song that is actually heard. The blind identification of audio tracks is commonly proceeded through fingerprinting techniques. The contributions in the field are numerous, both from the industrial actors [1][2] the academic world [3]. However, most audio fingerprinting methods are inefficient in the presence of several mixed tracks, and these techniques could only detect the presence of the tracks, not their respective gains. This article shows how a simple signal-based method answers this problem.

Our scope of interest is widened by considering the eventual presence of an additional unknown input in the mix process. Indeed, the estimation of known sources mix logically allows the deduction of the unknown source contribution. We will see that

the proposed system is able to extract this source from the output, and give an extensive study on the integrity of the signal extracted. This issue is indeed relevant in our use-case since the speakers microphones are usually directly connected to the mixing console with no possibility to retrieve the signal independently, while others pre-recorded sources are directly accessible to a program.

The definition of the mix estimation problem and the proposed algorithm are presented in Section 2, followed by the description of the experimental protocol of our study in Section 3. An analysis of the results and refining of the parameters will follow in section 4, and Section 5 concludes this work.

2. MIX ESTIMATION

2.1. Definition of the problem

The problem stated is the estimation of the fader gains of a mixing console from the known inputs and output. The inputs of the console are fed with pre-recorded sounds, e.g. jingles, liners or musical tracks. The output is directly retrieved from the mixing console. An important issue, is the effect of the track filters (modelled as Finite Impulse Response filters) applied on each input of the mixing console. The inputs considered in the estimation process are thus previously filtered with the corresponding impulse response, that is measured using Golay codes [4] on each input. The two objectives are the estimation of the fader gains in a dynamic context, and the estimation of an unknown additional input, that contains a signal that is not directly retrievable.

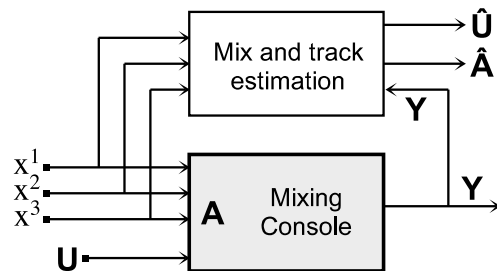


Figure 1: Architecture of the system.

The system architecture is summed up in figure 1. The following notations are used in the remainder of this article:

$\mathbf{X}_n^i = [x^i(n), \dots, x^i(n - N + 1)]^T$ is the N sample column vector for the i th input at instant n ,

$\mathbf{X}_n = [\mathbf{X}_n^1, \dots, \mathbf{X}_n^I]$ is the input matrix a scenario involving I known inputs, at instant n ,

$\mathbf{Y}_n = [y(n) \dots y(n - N + 1)]^T$ is the output column vector.

\mathbf{U}_n is the additional unknown voice input vector, at instant n ,

* This work was done during my PhD period at the radio station RTL.

$\mathbf{A}_n = [a_n^1, \dots, a_n^I]^T$ models the fader gain values at instant n . The mixing console effect is modeled by $\mathbf{Y}_n = \mathbf{X}_n \mathbf{A}_n + \mathbf{U}_n$.

2.2. Algorithm

The mix estimation is solved with least mean squares. \mathbf{A}_n is considered as the projection of the output \mathbf{Y}_n on the space generated by the input matrix \mathbf{X}_n . Let \mathbf{X}_n^\dagger denote the pseudo-inverse of \mathbf{X}_n , then:

$$\hat{\mathbf{A}}_n = \mathbf{X}_n^\dagger \mathbf{Y}_n = (\mathbf{X}_n^T \mathbf{X}_n)^{-1} \mathbf{X}_n^T \mathbf{Y}_n \quad (1)$$

The faders gain vector $\hat{\mathbf{A}}_n$ is estimated on frames of N samples, with a hop size of R samples between frames. The delay induced by the mixing process in the output can be rendered by the RIF filters applied to each input, and is thus ignored in our formalization.

The fader gain for each track i is thus described by a sequence $\mathbf{A}^i = [a_0^i, a_R^i, \dots, a_{k \cdot R}^i]$, sampled at $1/R$. The upper Figure 2 shows an example of estimated gain sequences $(\hat{\mathbf{A}}^i)_{i=1, \dots, I}$ for an added noise of 20 dB Signal to Noise Ratio, that models the \mathbf{U}_n signal. To reduce the distortion induced by the added noise, a post-process consisting of a median filter on F samples, is applied. Median filtering is a robust, fast, and very common way to smoothen estimation curves [5]. The lower Figure 2 illustrates the drastic effect in the estimates. The choice of the filter size, fixed to $F = 20$ samples in the figure, must meet a compromise in the reduction of distortions between static and transient parts.

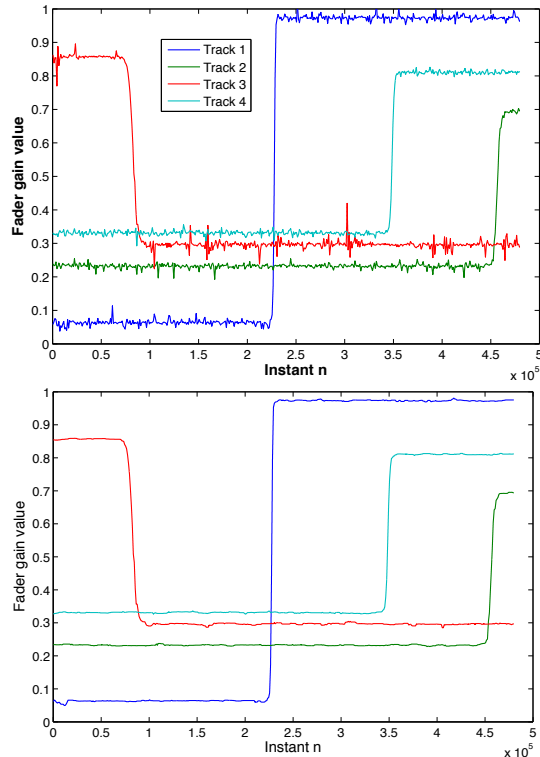


Figure 2: Estimated $\hat{\mathbf{A}}^i$ sequences with added noise 20 dB SNR, with $N = 2000$ and $R = 500$. (up) no post-processing (down) median filter, 20 sample window.

The unknown track is then estimated, with the estimated mix gain:

$$\hat{\mathbf{U}}_n = \mathbf{Y}_n - \mathbf{X}_n \hat{\mathbf{A}}_n \quad (2)$$

The remainder of this article focuses on the influence of parameters N , R and F on the estimation, for different Mix to Noise Ratios (MNR) or Mix to Added voice Ratio (MAR), where Mix denotes the mix of the known inputs : $\mathbf{X}_n \mathbf{A}_n$.

3. EXPERIMENTS

3.1. Corpus

The evaluation corpus consists of excerpts of radio broadcast news shows, and thus mainly filled with speech, with a possible background liner. The audio tracks are monophonic with 16 bits quantization, sampled at 16 kHz. Each result is computed on a 20 minutes mix simulation.

Four tracks are used for the known inputs \mathbf{X}^i ($I = 4$). The additional signal \mathbf{U} is either a Gaussian noise or another excerpt of the broadcast news. Since different speech signals are more correlated than music and speech signals, our experiment is more constrained than the original requisites. We have also tested the unknown track extraction with the musical known inputs from the RWC music genre database [6], but this brings no significant improvement.

3.2. Fade simulation

As stated earlier, the mix estimation process behaves differently on static and transient parts of the fader gain sequences (\mathbf{A}^i) . Indeed, the correct estimation of the transient is only done through a linear interpolation between successive frame values. The gain values are thus prone to more distortions on fadings.

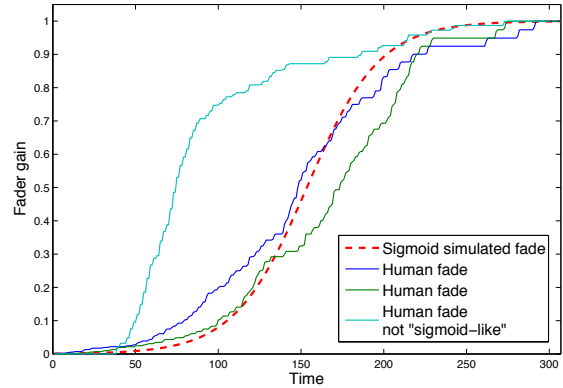


Figure 3: Example of measured fade curves (solid) and modeling by a sigmoid curve (dashed)

The way humans move faders is quite variable, as shown in the solid curve examples of Figure 3, acquired from a mixing console. However, the first two human fadings are quite similar to a sigmoid curve, defined by $S(t) = 1/(1 + e^{-\alpha t})$. This model is used here for the artificial fading transitions. Because of the fast convergence on the edges, it helps modelling a fast and continuous transition between two values.

In this experiment, the four tracks gain are changed alternatively at random intervals (around 15 s) and follow a sigmoid fade

curve during a random interval around 0.5 s. The mean duration of the total fade intervals on each 20 minute mix test is 1 minute.

3.3. Evaluation

For the evaluation of the mix estimation process, the criterion is the mean gain distortion on all tracks (expressed in dB) :

$$G_{dist} = 10 \log_{10} \frac{1}{I} \sum_i \frac{\|\hat{\mathbf{A}}^i - \mathbf{A}^i\|^2}{\|\mathbf{A}^i\|^2} \quad (3)$$

Since this problem is also a source separation problem (with high prior knowledge), the criteria presented in [7] for Blind Source Separation scoring are also relevant in this context, especially for the unknown track estimation. They give a more specific measure for separation than the usual Signal to Noise Ratio.

Let \mathbf{M} be the mixed signal without the unknown track \mathbf{U} , the estimate $\hat{\mathbf{U}}$ can be projected on \mathbf{U} and the mixed signal \mathbf{M} , with ϵ_{artif} the residual of the projection:

$$\hat{\mathbf{U}} = \langle \hat{\mathbf{U}}, \mathbf{U} \rangle \mathbf{U} + \langle \hat{\mathbf{U}}, \mathbf{M} \rangle \mathbf{M} + \epsilon_{artif}, \quad (4)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. The Signal to Distortion Ratio (SDR) is a global measure of the separation quality, while the Signal to Interference (SIR) and Signal to Artifacts (SAR) ratios respectively measure the amount of unknown track and artefacts remaining in the separated mixed signal. They are defined as:

$$SDR = 10 \log_{10} \frac{\|\langle \hat{\mathbf{U}}, \mathbf{U} \rangle \mathbf{U}\|^2}{\|\langle \hat{\mathbf{U}}, \mathbf{M} \rangle \mathbf{M} + \epsilon_{artif}\|^2} \quad (5)$$

$$SIR = 10 \log_{10} \frac{\|\langle \hat{\mathbf{U}}, \mathbf{U} \rangle \mathbf{U}\|^2}{\|\langle \hat{\mathbf{U}}, \mathbf{M} \rangle \mathbf{M}\|^2} \quad (6)$$

$$SAR = 10 \log_{10} \frac{\|\langle \hat{\mathbf{U}}, \mathbf{U} \rangle \mathbf{U} + \langle \hat{\mathbf{U}}, \mathbf{M} \rangle \mathbf{M}\|^2}{\|\epsilon_{artif}\|^2} \quad (7)$$

The *SIR* helps particularly in measuring the proportion of mixed signal kept in the estimation of the unknown track.

The same criteria are also defined on the restriction to the parts containing fades (see section 3.2): G_{dist}^F , SDR^F , SIR^F , SAR^F .

4. RESULTS

4.1. Mix without unknown input

Table 1 shows the gain distortion G_{dist} in the unnoised situation (i.e. $\mathbf{U}_n = 0 \forall n$) for different median filter length (F) and window size (N) values. Not surprisingly, the mix estimation is more accurate when the median filter is longer and the window more narrow. A negligible gain distortion of -62 dB is measured for the best case ($F = 100$ and $N = 50$). When restricted to fade intervals, G_{dist}^F is a few dB higher for all values of F and N but still remains very low in the best case ($G_{dist}^F = -58$ dB).

4.2. Robustness to distortions

Naturally, the gain distortion increases when noise is introduced in the mixed signal, and the parameters effect is different. The upper Figure 4 shows the gain distortion measured with F varying from 0 (no filtering) to 100, and N between 50 and 8000, for a SNR of 10dB. The figure clearly shows the correlation between the two

filt / N	50	200	500	2000	8000
0	-39.5	-34.3	-28.8	-27.9	-15.3
5	-45.1	-41.8	-34.1	-29.8	-15.3
15	-51.8	-52.1	-44.0	-29.8	-14.4
50	-58.9	-54.0	-44.0	-29.2	0.4
100	-62.5	-54.0	-44.0	-7.9	7.3

Table 1: Gain distortion G_{dist} for different configurations of F and N , without unknown input.

parameters optimal values: the minimal gain distortion G_{dist} remains stable when the product $F \cdot N$ is constant. Indeed, Eq. 1 gets more over-determined when N increases, and F must consequently be lowered to avoid over-smoothing of the gain curves. A global minimum is observed around $N = 2000$ and $F = 15$, with $G_{dist} = -20.0$ dB.

On the contrary, the gain distortion on the sole transitions (Figure 4) show a much more localized minimum. The minimum peak is also reached for $N = 2000$ and $F = 15$ with $G_{dist}^F = -15.9$ dB, but decreases strongly outside these values, even when keeping $F \cdot N$ constant. This shows the higher sensitiveness of the gain estimation on fadings.

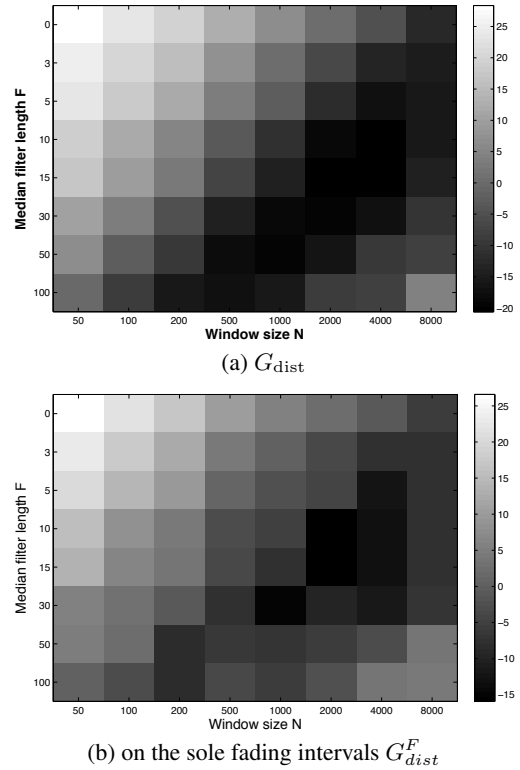


Figure 4: Gain distortion for different F and N values, with an added noise of 10dB SNR.

The same experiment is followed for 20 dB and 5 dB SNRs. Figure 5 compares G_{dist} (solid) and G_{dist}^F (dashed) for these three SNR values, with different F and a window length of $N = 2000$ samples. G_{dist} is minimized in most cases for $F = 15$. For short median filter lengths, the gain distortion is lower on fading intervals than on the whole signal for SNR of 20dB and 10dB. This re-

veals the distortion induced in the fading gain by over-smoothing.

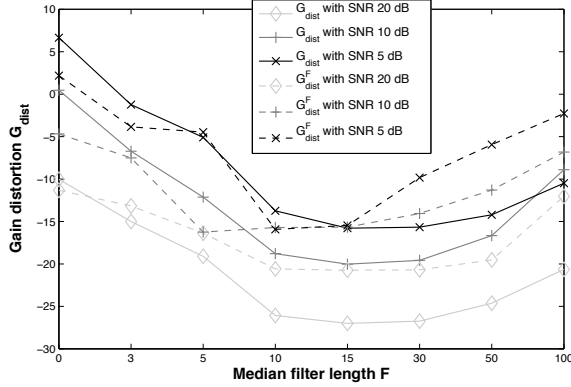


Figure 5: Evolution of G_{dist} (solid) and G_{dist}^F (dashed) with the median filter length ($N = 2000$), for different SNR values.

4.3. Unknown input estimation

In this next experiment, the scope of evaluation is restricted to the fading intervals. The previous experiment has provided some clues to calibrate F and N . If this noise is replaced by a speech track, the minimal gain distortion differs only by a few dB, and is still observed in most cases for $N = 2000$, as shown in table 2(a), where each column sums up the optimal configuration for a given Mix to Additional track Ratio (i.e. $\text{MAR} = \|\mathbf{M}\|^2 / \|\mathbf{U}\|^2$). The gain estimation is evaluated for different values of MAR ranging from 20 dB to -5 dB. For low MAR values (i.e. a stronger added signal) the gain distortion is much higher, and reaches -7dB in the best case for a -5dB MAR. The best configuration is clearly a median filter length of 30 samples and a window of $N = 2000$.

	MAR (dB)	20	10	5	0	-5
F		30	50	30	30	30
N		1000	500	2000	2000	2000
G_{dist}^F (dB)		-22.3	-16.0	-13.8	-10.0	-7.0
(a) Optimal N , F and gain distortion G_{dist}^F						
	MAR (dB)	20	10	5	0	-5
F		5	5	10	5	5
N		500	2000	500	500	500
SIR^F (dB)		48.9	51.8	51.4	50.1	56.0
(b) Optimal N , F and Signal to Interference Ratio SIR^F						
	MAR (dB)	20	10	5	0	-5
F		10	10	15	10	10
N		1000	1000	1000	1000	2000
SAR^F (dB)		18.2	22.9	24.3	25.1	26.8
(c) Optimal N , F and Signal to Artefacts Ratio SAR^F						

Table 2: Optimal values on fading intervals for different MAR.

The Signal to Interference Ratio, presented above, evaluates the separation of the unknown track \mathbf{U} by quantifying the proportion of the mixed signal \mathbf{M} present in the estimation $\hat{\mathbf{U}}$. N varies from 500 to 4000, and the median filter length F between 5 and 50. The SIR^F criterion on the sole fading intervals helps judging the separation capability for the different configurations. Table 2(b) shows, for each MAR value, the optimal N and F values, along with the maximum SIR^F . The latter criterion increases when the MAR gets higher, which shows that the G_{dist} criterion is

not relevant in evaluating source separation since it has an opposite behaviour. The SIR^F is maximized to 56dB with -5dB MAR.

Nevertheless, the artefacts are a much important part of the in noise induced in the source separation, than the interference. Table 2(c) shows the optimal Signal to Artefacts Ratio measured in the same experiment. The latter increases as well when the unknown track energy increases, and reaches 26.8 dB for a -5dB MAR, with $F = 10$ and $N = 2000$. Since the SAR^F is 30 dB lower than the SIR^F , the latter is considered negligible, and the global distortion measure SDR^F is considered equal to SAR^F . The optimal N and F are thus very close to the values estimated in Section 4.2 above.

A last study is done on the step length R . For each MAR value, the SAR^F score is measured for $R \in [\frac{N}{8}, \frac{N}{4}, \frac{N}{2}]$. A systematic improvement is observed with $R = \frac{N}{8}$ and $F' = 4F$. Table 3 shows the gain measured on the SAR^F evaluation criterion, when compared to $R = N$ and $F' = F$.

MAR (dB)	20	10	5	0	-5
SAR^F (dB)	24.3	26.5	27.4	28.4	28.9
ΔSAR^F (dB)	6.1	3.6	3.1	3.3	2.1

Table 3: Gain on the Signal to Interference Ratio on fadings SAR^F with a window hop of $R = \frac{N}{8}$ and $F' = 4F$ (where F is the optimal value with $R = N$) for different MAR values.

5. CONCLUSION

We have presented here an efficient and very simple algorithm for the estimation of a mix with the prior knowledge of the input and output signals. The optimal gain distortion is -20dB on the whole signal and -16dB on the gain fading transitions. The extraction of an added unknown track has shown very reliable since the global signal to distortion measured on the estimation reaches 28.9dB, this distortion is mostly due to artefacts induced by the algorithm.

The major weakness of our algorithm, though, lies in the need of a prior knowledge of the filters applied on each track by the mixing console. An interesting perspective would be the dynamic estimation of the filters response coupled with the mix estimation.

6. REFERENCES

- [1] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. ISMIR '02*, October 13-17 2002.
- [2] A. Li-Chun Wang, "An industrial-strength audio search algorithm," in *Proc. ISMIR '03*, 2003.
- [3] M. Ramona and G. Peeters, "Audio identification based on spectral modeling of bark-bands energy and synchronisation through onset detection," in *Proc. ICASSP*, May 2011.
- [4] S. Foster, "Impulse response measurement using golay codes," in *Proc. ICASSP '86*, April 1986, vol. 11, pp. 929-932.
- [5] P. F. Welleman, "Robust nonlinear data smoothers: Definitions and recommendations," in *Proc. National Academy of Sciences '77*, February 1977, vol. 74, pp. 434-436.
- [6] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Music genre database and musical instrument sound database," in *Proc. ISMIR*, October 2003, pp. 229-230.
- [7] R. Gribonval, L. Benaroya, E. Vincent, and C. Févotte, "Proposals for performance measurement in source separation," in *Proc. ICA*, April 1-4 2003, pp. 763-768.

PVSOLA: A PHASE VOCODER WITH SYNCHRONIZED OVERLAP-ADD

Alexis Moinet

TCTS Lab.

Faculté polytechnique

University of Mons, Belgium

alexis.moinet@umons.ac.be

Thierry Dutoit

TCTS Lab.

Faculté polytechnique

University of Mons, Belgium

thierry.dutoit@umons.ac.be

ABSTRACT

In this paper we present an original method mixing temporal and spectral processing to reduce the phasiness in the phase vocoder. Phasiness is an inherent artifact of the phase vocoder that appears when a sound is slowed down. The audio is perceived as muffled, reverberant and/or moving away from the microphone. This is due to the loss of coherence between the phases across the bins of the Short-Term Fourier Transform over time. Here the phase vocoder is used almost as usual, except that its phases are regularly reset in order to keep them coherent. Phase reset consists in using a frame from the input signal for synthesis without modifying it. The position of that frame in the output audio is adjusted using cross-correlation, as is done in many temporal time-stretching methods. The method is compared with three state-of-the-art algorithms. The results show a significant improvement over existing processes although some test samples present artifacts.

1. INTRODUCTION

Time-stretching of an audio signal is a process that increases or reduces the length of the signal while preserving its acoustic quality. In other words it reduces or increases the playback speed of the sound without changing its perceived content, as opposed to a change of the sampling frequency that causes a downward or upward frequency shift.

Many algorithms have been developed to achieve such a transformation. They generally belong to one of three categories [1]: time-domain, frequency-domain and model-based algorithms, although some methods combine several approaches (time and frequency, frequency and model).

Time-domain methods such as SOLA (*synchronized overlap-add*), WSOLA (*waveform similarity-based synchronized overlap-add*), SOLAFS (*synchronized overlap-add, fixed synthesis*), TD-PSOLA (*time-domain pitch-synchronous overlap-add*) [2, 3, 4] and their variants are usually applied to monophonic signals, for instance speech and singing recordings. The basic principle of these methods is to segment the signal into overlapping *frames* (i.e. blocks of consecutive audio samples) and either duplicate (drop) some frames or increase (reduce) the shift between each frame, in order to extend (compress) the duration of the signal.

Frequency or spectral-domain algorithms are most often based on the phase vocoder [5]. Compared to time-domain approaches, the phase vocoder has the advantage to work with both mono and polyphonic signals. Besides it theoretically overlaps frames perfectly in phase with each other. However in practice it produces a

sound that can be perceived as muffled, reverberant and/or moving away from the microphone [6, 7]. This distortion is called *phasiness* [8] and the accepted explanation for its presence is a loss of coherence between the phases across the bins of the Short-Term Fourier Transform over time, also called loss of vertical phase coherence. Different methods have been proposed in order to attenuate this artifact in [6, 7, 9].

Model-based approaches transform the audio signal into a set of frame-adaptive parameters that are decimated or interpolated to synthesize a time-scaled version of the sound. Linear Prediction-based analysis/synthesis, Harmonic plus Noise Model [10], Spectral Modeling Synthesis [11] and Sine + Transient + Noise Model [12] are good examples.

Some methods combine several approaches, as an enhanced version of SOLA [13] where a phase vocoder is used to modify the phases of each frame so that they overlap properly instead of adapting their position in the output audio signal. Another example is [14] which concatenates groups of time-domain frames with groups of frames generated by the phase vocoder. Besides STRAIGHT [15] could be considered as a mixed method to a certain extent.

In this paper we propose a new approach where a SOLA-like algorithm is used to periodically adapt the position of some frames in a phase vocoder (as opposed to using a phase vocoder to adapt the frames of SOLA in [13]). These frames are analysis frames used without phase modification which in turn causes a phase reset of the vocoder. This reduces the phasiness observed in audio signals without requiring any phase locking. We named this method PVSOLA (*Phase Vocoder with Synchronized Overlap Add*).

Phase reset or time-domain frame insertion has already been introduced by Karrer [16], Röbel [17] and Doran et al. [14]. Karrer resets the phases of the vocoder during silent parts, so that the distortion that it might cause is inaudible. Röbel preserves the transient components of a signal by resetting the phase-vocoder whenever a transient event is detected. Doran et al. do not abruptly reset the vocoder, instead they progressively alter the phases of the synthesis frames in order to regain coherency with the input signal. When the output and input signal become eventually in phase, a group of frames from the input is directly inserted in the output which is equivalent to a reset of the phase vocoder.

We review the principle of an STFT-based phase vocoder in Section 2 with the description of two possible approaches and different phase locking methods. Then we introduce an implementation of our method in Section 3 and we discuss its results and future developments in Sections 4 and 5.

This work is supported by a public-private partnership between University of Mons and EVS Broadcast Equipment SA, Belgium.

2. PHASE VOCODER

The underlying hypothesis of the phase vocoder is that a signal $x(n)$, sampled at frequency F_s , is a sum of P sinusoids, called *partials* [18]:

$$x(n) = \sum_{i=1}^P A_i \cos\left(\frac{n}{F_s} \omega_i + \phi_i\right) \quad (1)$$

each with its own angular frequency ω_i , amplitude A_i and phase ϕ_i . These 3 parameters are presumed to vary relatively slowly over time so that the signal is quasi-stationary and pseudo-periodic (e.g. speech and music). By segmenting the signal into overlapping frames to compute a Short-Term Fourier Transform (STFT), it is possible to use and modify the spectral amplitude and phase of each frame to either time-shift them (Section 2.1) or to interpolate new frames from them (Section 2.2).

2.1. Frame shifting

The most common implementation of the phase vocoder found in the literature [5, 7, 18] uses different sizes for the shift between frames (*hopsize*) during the analysis and the synthesis steps. The ratio between these two hopsizes equals the desired slow-down/speed-up factor. This means that to change the speed by a factor α with a synthesis hopsize R_s the analysis hopsize R_a must be:

$$R_a = \alpha R_s \quad (2)$$

Since the relative position of each frame in the output signal is different from that of the frames in the input signal, a simple overlap-add of the frames to generate that output will cause phase discontinuities. The main idea behind the phase vocoder is to adapt the phase of each partial according to the new hopsize R_s so that all the frames overlap seamlessly. Roughly speaking the adaptation needs to keep constant the variation of phase over time.

For each bin k of the STFT the phase variation between input frames i and $i-1$ is compared to the expected phase variation for that bin (a function of k and R_a). The difference between these two values (the *heterodyned phase increment*) is converted to the range $\pm\pi$ (Equation 6), divided by α and added to the theoretical phase variation for bin k in the output signal (a function of k and R_s). Finally this value is added to the phase of output frame $i-1$ to obtain the phase of output frame i (Equation 7). Note that the input frame 0 is reused as output frame 0 (Equation 3) and that the spectral amplitudes are not modified (Equation 4).

$$Y(0) = X(0) \quad (3)$$

$$|Y(i)| = |X(i)| \quad (4)$$

$$\Omega = \{0, \dots, k \frac{2\pi}{L}, \dots, (L-1) \frac{2\pi}{L}\} \quad (5)$$

$$\Delta\phi(i) = [\angle X(i) - \angle X(i-1) - R_a \Omega]_{2\pi} \quad (6)$$

$$\angle Y(i) = \angle Y(i-1) + R_s (\Omega + \frac{\Delta\phi(i)}{R_a}) \quad (7)$$

where $X(i)$ and $Y(i)$ are the Discrete Fourier Transforms (DFT) of the i^{th} input and output frames. $X(i)$, $Y(i)$, Ω and $\Delta\phi(i)$ are L -sample vectors with L the length of a frame. $[\cdot]_{2\pi}$ denotes the conversion of the phase to the range $\pm\pi$ [18].

Once the DFT of a frame has been calculated the synthesis frame samples are computed by Inverse Discrete Fourier Transform (IDFT) and the frame is added by overlap-add to the output signal.

2.2. Frame generation

Another implementation of the phase vocoder was proposed by Dan Ellis in [19]. Contrary to the previous method it uses the same hopsize between the frames at analysis and synthesis time. Obviously when doing time-stretching the number of frames used to synthesize the output is different from the number of frames extracted from the input. Frames have to be dropped or created one way or another. In the algorithm developed by Ellis all frames are generated by interpolating the spectral amplitudes and accumulating the phase variations between the analysis frames.

The first step sets the initial synthesis frame spectrum $Y(0)$ equal to the initial analysis frame spectrum $X(0)$:

$$|Y(0)| = |X(0)| \quad (8)$$

$$\angle Y(0) = \angle X(0) \quad (9)$$

For the following synthesis frames the synthesis frame indices j are linearly mapped to the analysis indices i using Equation 10:

$$i = \alpha j \quad (10)$$

where i is generally not an integer value. For instance if the speed factor α is 0.5 ($2\times$ slower), $Y(7)$ corresponds to a frame position in the original audio equal to $\alpha \times 7 = 3.5$ (i.e. located between $X(3)$ and $X(4)$).

The spectrum $Y(j)$ of the j^{th} synthesis frame is a function of the amplitude and phase variations of its “surrounding” analysis frames as well as $\angle Y(j-1)$:

$$\lambda = i - [i] \quad (11)$$

$$|Y(j)| = (1 - \lambda)|X([i])| + \lambda|X([i] + 1)| \quad (12)$$

$$\Delta\phi(i) = [\angle X([i] + 1) - \angle X([i])]_{2\pi} \quad (13)$$

$$\angle Y(j) = \angle Y(j-1) + \Delta\phi(i) \quad (14)$$

where $[i]$ is the integer value of i (the largest integer not greater than i). Finally the IFFT of each $Y(j)$ is computed and the samples are overlap-added into the output signal.

2.3. Phase locking

The methods presented in Section 2.1 and 2.2 are applied independently to each bin k of the spectrum in order to keep intact the phase constraints along the time (or horizontal) axis of the spectrogram. As a consequence there is no constraints with regard to the vertical axis: if there is a dependency between bins $k-1$, k , and $k+1$ in the input signal it is lost in the process. This causes the apparition of the phasiness artifact [8].

In order to correct this problem several algorithms have been proposed. In [6] Puckette uses the phase of the sum of the spectral values from bins $k-1$, k , and $k+1$ as the final phase value $\angle Y^*(i)$ for bin k :

$$\angle Y_k^*(i) = \angle(Y_{k-1}(i) + Y_k(i) + Y_{k+1}(i)) \quad (15)$$

Laroche et al. [7] proposed a somewhat more complex approach: the peaks in the spectrum are detected and the phases of their corresponding bins are updated as usual by the phase vocoder. The other bins located in the region of influence of each peak have their phases modified so as to keep constant their phase deviation from the peak's phase. As a result there is a horizontal phase locking for the peaks and a vertical phase locking for all the other parts

of the spectrum. A refinement of this method is to track the trajectories of the peaks over time and use the previous phase of each peak to compute the new one. This is important if a peak changes from one bin to another to avoid its phase being based on the phase of a previous non-peak bin. However tracking peaks over time is not always straightforward (peaks can appear, disappear, split or merge which increases the complexity of the task).

For small lengthening ratio Dorran et al. [14] recover phase coherence by slightly adjusting the phase of each synthesis frames so that after a few frames it converges to an almost perfect overlap with the analysis frame. From that point on a group of frames from the original signal can be added directly to the output signal without any phase transformation and therefore resulting in a (locally) perfect-quality audio signal. The phase gradual adjustment is calculated in order to be perceptually undetectable by a human ear.

3. PVSOLA

The new method presented in this section comes from an experimental observation we made on the phase vocoder (using [19]) and on the phase-locked vocoder (using *Identity Phase Locking* [7] as implemented in [18]):

Phasiness in the vocoder does not appear (or is not perceived) immediately. It takes a few frames before becoming noticeable.

A simple experiment to observe this phenomenon is to alter a phase-locked vocoder so that the phase-locking happens only once every C frame. The other frames are processed with a normal phase vocoder. For small values of C (typically 3 to 5 frames), the difference in phasiness with a fully locked signal is barely noticeable at all (some artifact/ripples may appear in the spectrogram though). For larger values of C phasiness becomes audible in the vocoder output. We propose the following explanation for this behavior: the loss of vertical coherence is a slow phenomenon, it is not instantaneous, and the spectral content also vary relatively slowly (hypothesis of quasi-stationarity in Section 2). Therefore every time a peak is detected and locked its neighboring bins undergo some kind of phase reset: their final phase is only a function of the change of the peak's phase and their phase difference relatively to the peak's original phase. As for the peak, since the signal varies slowly it can be assumed that its position remains more or less coherent from one frame to another (or even across 3 to 5 frames) even if it changes of bin (the bin change is never an important jump in frequency).

3.1. Method overview

Based on these observations we propose to combine a time-domain and a frequency-domain approach. The method consists in a periodic reset of a phase vocoder by copying a frame directly from the input into the output and using it as a new starting point for the vocoder. The insertion point for the frame in the output is chosen by means of a cross-correlation measure.

3.2. Implementation details

We propose the following framework: first we generate C synthesis frames (f_0, \dots, f_{C-1}) using a phase vocoder. Each frame f_i is

L -sample long and is inserted in the output signal by overlap-add at sample t_i with:

$$t_i = iR_s \quad (16)$$

where t_i is the position at which the first sample of the synthesis frame is inserted and R_s is the hopsize at synthesis (note that we choose $R_s = L/4$ as is usually done in the literature). The last frame generated (f_{C-1}) is inserted at position t_{C-1} , the next one (f_C) should be inserted at t_C . Now instead of another vocoded frame we want to insert a frame f^* extracted directly from the input audio in order to naturally reset the phase of the vocoder but we know that this would cause phase discontinuities.

In order to minimize such discontinuities we allow to shift the position of f^* around t_C in the range $t_C \pm T$ (T is called the *tolerance*). The shift is obtained by computing the cross-correlation between the samples already in the output and the samples of f^* . However some samples of the output are “incomplete”, they still need to be overlap-added with samples that would have been generated in the next steps of the phase vocoder (i.e. samples obtained by overlap-adding frames f_C, f_{C+1}, \dots). As a result a frame overlapped in another position than t_C would cause a discontinuity in the otherwise constant time-envelope of the time-scaled signal. Besides the cross-correlation would be biased toward negative shifts around t_C . To overcome these problems additional frames (f_C, f_{C+1}, \dots, f_F) are generated by the phase vocoder and temporarily inserted so that t_F respects the constraint in Equation 17:

$$t_F > t_C + L + T \quad (17)$$

which means that the first sample of the coming frame f_F would be inserted T samples after the end of f_C and that the output signal is “complete” up to sample t_F (no samples would be overlap-added anymore before that sample in a normal phase vocoder).

Position t_C corresponds to a position u_C in the input signal:

$$u_C = \alpha t_C \quad (18)$$

The next step consists in selecting a frame f^* of length L starting at sample u_C ¹ in the input signal and adding it in the output signal at position $t_C + \delta$ with $-T \leq \delta \leq T$ (we fixed the tolerance $T = 2R_s$). Equation 21 defines χ , a cross-correlation measure between the frame f^* (Equation 20) and the output samples o (Equation 19) already generated:

$$o = \{y(t_C - 2R_s), \dots, y(t_C + L - 1 + 2R_s)\} \quad (19)$$

$$f^* = \{x(u_C)h^2(0), \dots, x(u_C + L - 1)h^2(L - 1)\} \quad (20)$$

$$\chi = \text{xcorr}(o, f) \quad (21)$$

where $\{\}$ stands for a vector of values (a frame), $h^2(n)$ is the square of a Hann window (as defined in Equation 26) and xcorr is the cross-correlation function. $x(n)$ and $y(n)$ are the original and time-stretched signal respectively. The optimal value of δ corresponds to the position of the maximum of $|\chi_s|$, the subset of χ (as defined in Equation 23) that corresponds to an insertion of f^* in the position range $t_C \pm 2R_s$. Figure 1 shows an example of finding the offset δ using Equations 22 to 25:

$$\varepsilon = L + 4R_s = 2L \quad (22)$$

$$\chi_s = \{\chi(\varepsilon), \dots, \chi(\varepsilon + 4R_s)\} \quad (23)$$

$$p = \text{argmax}(|\chi_s|) \quad (24)$$

$$\delta = p - 2R_s \quad (25)$$

¹rounded to the nearest integer

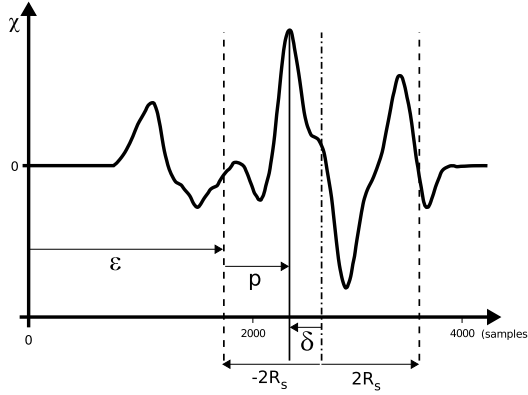


Figure 1: δ is computed from the position p of the maximum value of a subset of χ . The dashed lines delimit the subset χ_s and the dash-dotted line represents a positioning of f^* exactly at $t = t_c$. In this example δ is < 0 and $\chi_s(p) > 0$. The frame length L is 1024.

Notice that each frame processed through the phase vocoder undergoes two hann-windowing: one before the DFT and one after the IDFT before being overlap-added in the time-stretched signal. Therefore f^* has to be windowed by the square of a Hann window (Equation 20) in order to overlap-add properly with the output signal and the future frames. The Hann window $h(n)$ is defined as:

$$h(n) = \begin{cases} 0.5 - 0.5 \cos\left(\frac{2\pi n}{L}\right) & \text{if } n = 0, \dots, L-1 \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

This definition is slightly different from the definition usually encountered (the denominator in the fraction is L instead of $L-1$) for the cumulated windowing would present a small ripple otherwise as explained in [20].

Then f^* is multiplied by the sign of $\chi_s(p)$ (in case of a negative peak) and overlap-added to the output audio (Figure 2).

Before inserting f^* the output samples between $t_c + \delta$ and $t_c + \delta + L - 1$ are windowed by a function $w(n)$ so that the overall accumulated windowing of the output remains constant (taking into account the frames yet to come). This also means that the samples of the output signal beyond $t_c + \delta + L - R_s$ that have been generated to compute the cross-correlation are set to zero. The computation of the envelope $w(n)$ applied to the time-stretched signal is presented in Figure 3 and Equation 27:

$$w(n) = h^2(n + 3R_s) + h^2(n + 2R_s) + h^2(n + R_s) \quad (27)$$

Finally since the frame f^* has been inserted “as is” the phase vocoder can be reinitialized to start a new step of the time-scaling process as if f^* were its initial frame f_0 and $t_c + \delta$ were its initial time position t_0 . Note that each analysis frame used during this new step must be inverted if $\chi_s(p) < 0$.

3.3. Discussion

It is important to notice that due to the accumulation of shifts δ (one for each iteration) a drift from the original speed factor α

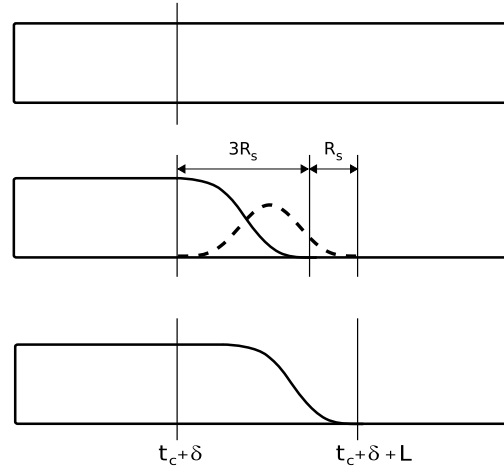


Figure 2: Schematic view of the insertion of a frame f^* at position $t_c + \delta$. Top: output signal after insertion of additional frames for cross-correlation computation. Middle: windowed output signal (solid line) and frame f^* windowed by the square of a Hann window (dashed line). Bottom: resulting signal before the next iteration. The upcoming windowed frames will add to a constant time-envelope with this signal.

could occur if no measure is taken to correct it. In our implementation we sum the values of δ for each phase reset and obtain a drift Δ . When Δ exceeds $\pm R_s$ the number of frames synthesized in the next iteration will be $C \mp 1$ and the value of Δ will change to $\Delta \mp R_s$. Theoretically Δ could even exceeds $\pm 2R_s$, in which case the number of frames synthesized will be $C \mp 2$ and Δ will become $\Delta \mp 2R_s$.

Another interesting fact is that if we set $C = 0$, the resulting algorithm is very close to a SOLA-like method except that the additional frames used for the cross-correlation are still generated by a phase vocoder. On the contrary $C = \infty$ changes the method back into a non-locked phase vocoder.

Finally in Section 3.2 we take the first sample of a frame as the reference for positioning. One might use the middle sample of each frame instead. This will not create any significant difference with the method proposed above.

4. RESULTS

This method can be applied to any phase-vocoder algorithm. For the following tests we implemented a modified version of the algorithm from [19]. We performed both formal and informal assessments presented respectively in Section 4.1 and 4.2.

4.1. Formal listening tests

We use sentences selected from the CMU ARCTIC databases [21] among the 4 US speakers, namely *clb*, *slt*, *bdl* and *rms* (two female and two male speakers). 50 sentences are randomly picked for each speaker and each sentence is processed by 4 different algorithms: a phase-vocoder, a phase-locked vocoder, a time-domain method (SOLA) and our method PVSOLA. Each process is applied with two speed factors: $\alpha = 1/1.5$ and $\alpha = 1/3$ (i.e. 1.5 and 3 times slower).

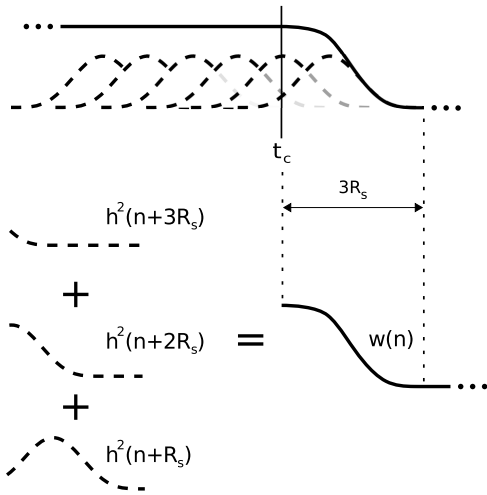


Figure 3: Schematic view of the computation process for the weighting function $w(n)$ that will be applied to the output signal after $t_c + \delta$. Top: in a standard phase vocoder, the squared Hann windows would sum to a constant value except for the last samples because there are frames not yet overlap-added after t_c . We want to reproduce that behavior at $t_c + \delta$ so that f^* overlap-adds seamlessly. Bottom: The time envelope is the sum of three squared Hann windows with a shift R_s between each one.

For the two phase vocoders we use the implementation available in [18] and for SOLAFS we use the implementation from [22]. We empirically set $L = 512$ samples and $R_s = L/4$ for the vocoders and PVSOLA. In our informal tests SOLAFS generally provided better quality with $L = 256$ so we kept that value. The parameters specific to PVSOLA are $C = 3$ and $T = 2R_s$.

PVSOLA is compared to the other three methods via a *Comparative Mean Opinion Score* (CMOS) test [23]. Participants are given the unprocessed audio signal as a reference (R) and they are asked to score the comparative quality of two time-stretched versions of the signal (both of them with the same speed modification). One is PVSOLA, the other is randomly chosen among the three state-of-the-art algorithms. The two signals are randomly presented as A and B. Each listener takes 30 tests, 10 for each concurrent method. The question asked is: “When compared to reference R, A is: much better, better, slightly better, about the same, slightly worse, worse, much worse than B?”

Each choice made by a listener corresponds to a score between ± 3 . In case A is PVSOLA, “much better” is worth 3 points, “better” 2 points and so on until “much worse” which means -3 points. On the contrary when B is PVSOLA, the scale is reversed with “much worse” worth 3 points and “much better” -3 points. In short when PVSOLA is preferred it gets a positive grade and when it is not it gets a negative one. 16 people took the test (among which 9 are working in speech processing) and the results are shown in Table 1 and Figure 4 and 5.

From these results one can see that for a speed slowdown factor of 1.5 our method is globally preferred except for SOLAFS with female voices where both methods are deemed equivalent. Besides SOLAFS performs relatively better than the phase-locked vocoder which in turn performs better than the phase vocoder. This is an expected result as time-domain methods usually give better results when applied to speech and the phase-locked vocoder is

Table 1: CMOS test results with 0.95 confidence intervals for female (clb and slt) and male (bdl and rms) speakers. PVSOLA is compared to the phase vocoder (pvoc), the phase-locked vocoder (plock) and SOLAFS.

$1/\alpha$	female	
	1.5	3
pvoc	2.03 ± 0.3	0.66 ± 0.43
plock	0.97 ± 0.41	1.86 ± 0.3
solafs	0.14 ± 0.32	1.21 ± 0.27

$1/\alpha$	male	
	1.5	3
pvoc	2.49 ± 0.32	1.05 ± 0.47
plock	1.78 ± 0.29	1.71 ± 0.3
solafs	1.13 ± 0.36	1.77 ± 0.27

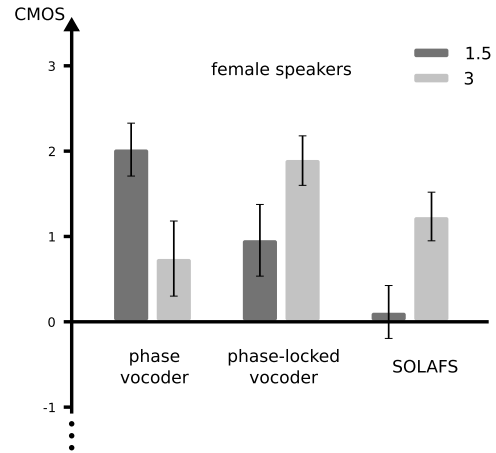


Figure 4: Results for the CMOS test for female speakers clb and slt. The dark and light gray bars represent the mean CMOS score for a speed ratio of respectively 1.5 and 3. 0.95 confidence intervals are indicated for information.

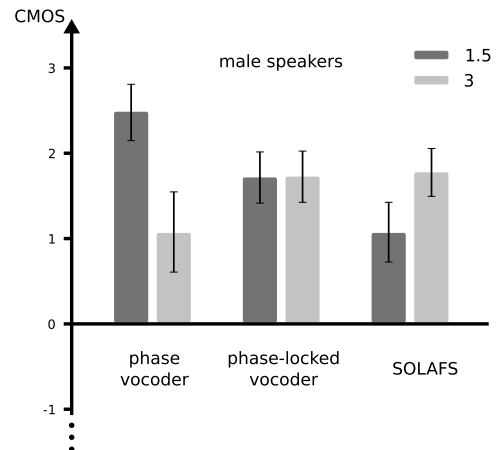


Figure 5: Results for the CMOS test for male speakers bdl and rms. The dark and light gray bars represent the mean CMOS score for a speed ratio of respectively 1.5 and 3. 0.95 confidence intervals are indicated for information.

supposed to be better than the phase vocoder.

For the higher slowdown factor 3, our method is again observed to outperform other approaches, notably better than SOLA in both tables and better than the phase-locked vocoder for female voices, but it has lost ground to the normal phase vocoder which has a better score than the two other approaches. After the test we discussed this with the listeners and we could establish that it was not a mistake. Indeed with this time-stretching ratio every method produces more artifacts (frame repetition for SOLA, metallic sound for the phase-locked vocoder, phasiness for the phase vocoder and some sort of amplitude modulations for PVSOLA). The listeners said that in some cases they “preferred” the defect of the phase vocoder to that of PVSOLA for a certain number of sentences of the dataset. It is still a minority of files for which this happens since the overall result is still in favor of PVSOLA but this has to be analyzed further.

4.2. Informal tests and discussions

We applied the algorithm to various signals: speech, singing voice, mono and polyphonic music and obtained improved results over all other methods for monophonic signals (speech, singing and music) while the algorithm suffers from audible phase mismatches for polyphonic signals.

Several values for C and L have been tried and the best trade-off seems to be $C = 3$ and $L = 512$ samples for a sampling frequency $F_s = 16$ kHz (i.e. $L = 32$ ms). As for other sampling frequencies (in singing and music data) we set L so that it also corresponds to about 30 ms. Nevertheless we noticed that in general the algorithm is not very sensitive to the value of L (between 20 and 40 ms). For $C = 3$ and a reasonable speed factor (between 1 and 3 times slower) we generally notice an important reduction of the phasiness. We generated some test samples for even slower speed factor ($\times 5$) with mixed results (some good, others presenting many artifacts).

For larger values of C perceptible phase incoherencies appear in the time-stretched signals probably because the phases of the different partials are already out-of-phase with each other. It seems that the cross-correlation measure can help to match some of these partials with the ones from the input frame f^* but not all of them thus creating artifacts that resemble an amplitude modulation (the audio sounds “hashed”, sometimes a beat appears at a frequency corresponding to CR_s). Note that even for values of $C \leq 3$ these mismatches may still appear but to a lesser extent, they are often almost inaudible. However discussions with listeners have shown that in some worst-case scenarios they can become a real inconvenience as explained in section 4.1.

As a side-effect of the algorithm, transients tend to be well-preserved contrary to what happens with time-domain (transient duplication) or phase vocoder-based algorithms (transient smearing). Apparently f^* can be advantageously positioned so that the transient is preserved due to the relatively large value of T . Although this may prove interesting it is not systematic and has yet to be investigated.

The main drawback of our method lies in its computational complexity when compared with time-domain or phase vocoder approaches. Indeed not only do we compute a cross-correlation every C frame but we also generate extra frames for its computation that will be eventually dropped and replaced by new ones. Roughly speaking we measured that our MATLAB implementation was three to four times slower than a phase vocoder. A pro-

file of the process shows that the most time-consuming task is by far the cross-correlation computation (about 40%). However results of benchmarking within MATLAB must always be taken with care since some operations (such as selecting a frame in a signal) are not well-optimized. We estimate that a C implementation of PVSOLA could be less than two times slower than that of a phase vocoder.

5. FUTURE WORK

We plan to work on different aspects of PVSOLA that can be improved:

- in [13] Röbel proposes to modify a cross-correlation to take into account only the partials and ignore the noisy components. We could use this method to refine the positioning of the frames f^* and reduce the artifacts of PVSOLA.
- For the moment we developed and implemented our algorithm as a SOLA-modified phase vocoder. A major change would be to use a WSOLA-like approach to the selection of f^* . Indeed we could select a frame from the input signal that would be optimal for an insertion at t_c instead of trying to find the best position $t_c + \delta$ for a given frame. This would suppress at the same time the need for additional frames (used for the cross-correlation computation) and for occasional additions or removals of frames when $|\Delta| > R_s$ (see Section 3.3). We are currently working on this topic.
- The results on polyphonic sounds are not as good as those on monophonic sounds. We plan to investigate this problem as well.
- PVSOLA has only been tested on a standard phase vocoder. Using a phase-locked vocoder could make it possible to increase the optimal value for C thus reducing the computational load.

6. CONCLUSIONS

This paper presented a new approach to modify the length of an audio signal without changing its perceived content. The method proposes a combination of a time-domain and a frequency-domain process. It consists in a periodic reset of a phase vocoder by copying a frame directly from the input into the output and using it as a new starting point for the vocoder. The insertion point for the frame in the output is chosen by means of a cross-correlation measure. Informal listening tests have highlighted a reduction of the phase vocoder’s phasiness and formal listening tests have shown that our method was generally preferred to existing state-of-the-art algorithms. Both formal and informal tests have pointed out that under certain circumstances the quality of the time-stretched audio could be perceived poorly because of discontinuities in the signal. Various suggestions have been made to improve this situation as part of future work or ongoing research.

7. EXTERNAL LINKS

Examples of audio time-stretching with PVSOLA are available at: <http://tcts.fpms.ac.be/~moinet/pvsola/>

8. REFERENCES

- [1] J. Bonada, “Audio time-scale modification in the context of professional audio post-production”, research work for PhD program, Universitat Pompeu Fabra, Barcelona, Fall 2002.
- [2] W. Verhelst, “Overlap-add methods for time-scaling of speech”, *Speech Communication*, vol. 30, no. 4, pp. 207–221, April 2000.
- [3] D. Hejna and B.R. Musicus, “The SOLAFS time-scale modification algorithm”, Tech. Rep., BBN Technical Report, July 1991.
- [4] E. Moulines, F. Charpentier, and C. Hamon, “A diphone synthesis system based on time-domain prosodic modifications of speech”, in *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Glasgow, Scotland, May 23-26 1989, pp. 238–241.
- [5] M. Dolson, “The phase vocoder: A tutorial”, *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, Winter 1986.
- [6] M. Puckette, “Phase-locked vocoder”, in *Proc. of IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, NY, USA, Oct. 15-18 1995, pp. 222–225.
- [7] J. Laroche and M. Dolson, “Improved phase vocoder time-scale modification of audio”, *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, May 1999.
- [8] J. Laroche and M. Dolson, “Phase-vocoder: about this phasiness business”, in *Proc. of 1997 IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA, Oct. 19-22 1997, pp. 55–58.
- [9] J. Bonada, “Automatic technique in frequency domain for near-lossless time-scale modification of audio”, in *Proc. of the International Computer Music Conference (ICMC)*, Berlin, Germany, 27 August – 1 September 2000, pp. 396–399.
- [10] Y. Stylianou, *Harmonic plus noise models for speech combined with statistical methods, for speech and speaker modifications*, Ph.D. thesis, École Nationale Supérieure des Télécommunications, 1996.
- [11] X. Serra and J. Bonada, “Sound transformations based on the sms high level attributes”, in *Proc. of the 1st International Conference on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, Nov. 19-21 1998.
- [12] T.S. Verma and T.H.Y. Meng, “Time scale modification using a sines+transients+noise signal model”, in *Proc. of the 1st International Conference on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, Nov. 19-21 1998, pp. 49–52.
- [13] A. Röbel, “A shape-invariant phase vocoder for speech transformation”, in *Proc. of the 13th International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10 2010.
- [14] D. Doran, E. Coyle, and R. Lawlor, “An efficient phasiness reduction technique for moderate audio time-scale modification”, in *Proc. of the 7th International Conference on Digital Audio Effects (DAFx-04)*, London, UK, Oct. 5-8 2004, pp. 83–88.
- [15] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigne, “Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds”, *Speech Communication*, vol. 27, no. 3-4, pp. 187–207, April 1999.
- [16] T. Karrer, E. Lee, and J. Borchers, “Phavorit: A phase vocoder for real-time interactive time-stretching”, in *Proc. of the International Computer Music Conference (ICMC)*, New Orleans, USA, Nov. 6-11 2006, pp. 708–715.
- [17] A. Röbel, “A new approach to transient processing in the phase vocoder”, in *Proc. of the 6th International Conference on Digital Audio Effects (DAFx-03)*, London, UK, Sept. 8-11 2003.
- [18] T. Dutoit and J. Laroche, *Applied Signal Processing – A Matlab-Based Proof of Concept*, chapter How does audio effects processor perform pitch shifting ?, pp. 149–185, Springer Science+Business Media, 2009.
- [19] D. P. W. Ellis, “A phase vocoder in Matlab”, 2002, Web resource, last consulted in March 2011.
- [20] A. De Götzen, N. Bernardini, and D. Arfib, “Traditional (?) implementations of a phase-vocoder: the tricks of the trade”, in *Proc. of the 3rd International Conference on Digital Audio Effects (DAFx-00)*, Verona, Italy, Dec. 7-9 2000, pp. 37–44.
- [21] John Kominek and Alan W Black, “CMU arctic databases for speech synthesis”, Tech. Rep., Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2003.
- [22] D. P. W. Ellis, “SOLAFS in Matlab”, 2006, Web resource, last consulted in March 2011.
- [23] V. Grancharov and W. Kleijn, *Handbook of Speech Processing*, chapter Speech Quality Assessment, pp. 83–99, Springer, 2007.

VIVOS VOCO: A SURVEY OF RECENT RESEARCH ON VOICE TRANSFORMATIONS AT IRCAM

P. Lanchantin, S. Farner, C. Veaux, G. Degottex,
N. Obin, G. Beller, F. Villavicencio, S. Huber,
G. Peeters, A. Roebel, X. Rodet*

Institut de Recherche et Coordination Acoustique/ Musique
1, place Igor Stravinsky
75004 Paris, France
lanchant@ircam.fr

ABSTRACT

IRCAM has a long experience in analysis, synthesis and transformation of voice. Natural voice transformations are of great interest for many applications and can be combine with text-to-speech system, leading to a powerful creation tool. We present research conducted at IRCAM on voice transformations for the last few years. Transformations can be achieved in a global way by modifying pitch, spectral envelope, durations etc. While it sacrifices the possibility to attain a specific target voice, the approach allows the production of new voices of a high degree of naturalness with different gender and age, modified vocal quality, or another speech style. These transformations can be applied in realtime using ircamTools TRAX. Transformation can also be done in a more specific way in order to transform a voice towards the voice of a target speaker. Finally, we present some recent research on the transformation of expressivity.

1. INTRODUCTION

Founded by Pierre Boulez in 1977, IRCAM, the Institute for Research and Coordination Acoustic / Music, is one of the world's largest public research centers dedicated to both musical expression and scientific research. It is a unique location where artistic sensibilities collide with scientific and technological innovation. It has extensive experience in analysis, transformation and synthesis of sounds and in particular of speech dating back to its beginnings [1, 2, 3, 4], and has continued until today. For the last years, research in speech was mostly oriented towards voice transformations and Text-to-Speech synthesis (TTS). For instance, IRCAM develops software SUPERVP that includes treatments specifically designed for speech [5, 6, 7, 8], the software DIPHONE STUDIO [9], which use the concept of acoustic units, PSOLA and SINOLA [10]. It recently proposed the ircamTools commercial plug-in TRAX which offers a novel approach to voice synthesis through gender and age sound transformations. IRCAM is also developing TTS systems by unit selection [12, 13, 14] and HMM-based speech synthesis [15, 16]. Other studies related to speech include the modeling of the prosody [17] and the independent modeling of glottal source and vocal tract which allows to treat them separately.

Voice transformations, TTS and their combined use have a great potential for artistic creations. Regarding the transformation of voice, IRCAM has worked on French films such as "Farinelli" by G. Corbiau, "Vatel" by R. Joffé, "Vercingétorix" by J. Dorfmann, "Tirésia" by B. Bonello, "Les Amours d'Astrée et de Céladon" by E. Rohmer, but also for "Jeu d'enfants" by Y. Samuël, "The Last Dragon" by M. Schultz. In the film industry, transformations can be useful to convert the actor's voice into another that is more suitable for the role, it can allow the use of one actor to achieve several voices in dubbing or animation movies, modification of accentuation of recorded speech, or creation of animal voices, to mention a few applications. In the music field, IRCAM has created a synthetic voice for the opera "The Mask of Orpheus" by H. Birtwistle using the software CHANT [18]. A real-time synthesis of spoken and sung choruses [19] has been developed in the Max software platform used particularly in the opera "K" by P. Manoury at Opera Bastille. IRCAM has also worked and still works on several projects including Multimedia voice. For instance, in "Les Variations Darwin", the stage director J. F. Peyret worked with IRCAM on an order of the Theatre National de Chailot. His project used the automatic generation of text and speech processing in real time, with music by A. Markeas. Similarly, IRCAM has worked with director E. Genovese on the transformation of voice actors from the "Comédie française" and sound environments for the staging of "le privilège des chemins" and with the composer J. Harvey on "Mortuos Plango, Vivos Voco" [20] and lastly on "Speaking".

In this paper, we will focus on voice transformations. However, for the interested reader, we note that recent research on HMM-based speech synthesis [15, 16] combined with advanced modeling of the prosody [17] are promising on a compositional perspective and will definitely be of interest for composers in the next years.

The paper is organized as follows, in Section 2, we introduce the basic tools that are used for transformations, in Section 3 we present the transformations of type and nature. In Section 4, we introduce the transformation towards a target voice. In Section 5, we present transformation of expressivity. Finally we conclude and give further directions in Section 6.

2. SIGNAL TRANSFORMATION

Acoustically, the vocal organ consists of the vocal tract (mouth and nose cavities) as a resonance chamber and the larynx (the vo-

* Part of the research presented in this paper was supported by FEDER Angelstudio : Générateur d'Avatars personnalisés ; 2009-2011

cal folds (glottis), the false vocal folds and epiglottis) as the principal sound producing mechanism, thus called the voice source. A model relating the physics of the voice and the emitted signal is necessary for changing the speech signal. A number of signal-centered methods have been developed, the most successful ones probably being the PSOLA method [21], harmonic plus noise methods (HNM) [22], STRAIGHT [23] and the phase vocoder [7, 8, 24]. While all these methods can perform transposition and time stretching of the signal, only the latter two principles allow finer modification of the signal in the frequency domain. SUPERVP - our improved version of the phase vocoder is under continuous development for musical applications at IRCAM, and serves as the engine of AUDIOSCULPT, a powerful graphically interactive software for music modification [26]. However the optimal model is probably one than separates the glottal source (as far as possible) from the vocal tract. In [27], we recently proposed a new model in which the glottal source is separated into two components: a deterministic glottal waveform Liljencrants-Fant (LF) model and a modulated Gaussian noise.

We first present the phase vocoder and the improvements that have been made in 2.1, then we introduce the glottal source modeling in 2.2 which will be the basic tools used for the different transformations described in the following of this paper.

2.1. The phase vocoder and improvements

The basic phase vocoder [28] is roughly a series of band filters, in practice implemented as successive Short-Time Fourier Transforms (STFTs), that reduce the signal into amplitudes and phases in a uniform time-frequency grid. Combined with resampling and changing of the time step between analysis and synthesis, this method allows for high-fidelity time stretching and pitch transposition as well as modification of the amplitude of each point in the grid, and thus an enormous potential for transformations.

A well-known artifact of the phase vocoder is the introduction of “phasiness”, in particular for speech, the result sounding strangely reverberant or with a lack of presence of the speaker. Improvements added to our implementation of the phase vocoder and constituting SUPERVP are: detection and processing of transients [25], waveform preservation for single-source processing [7, 8], robust spectral-envelope estimation [5], and dynamic voicing control based on spectral-peak classification [29].

2.2. Glottal source model

Recently, we proposed an new glottal source and vocal-tract separation method called Separation of Vocal-tract and Liljencrants-Fant model plus Noise (SVLN [27]). In this method, the glottal excitation is separated into two additive components: a deterministic glottal waveform modeled by the LF model [30] and a noise component modeled by a Gaussian noise. The parametrization by only two parameters - the shape parameter Rd and the noise level σ_g - allows an intuitive control of the voice quality. Rd characterizes the slope of the glottal spectrum and can be seen as a measure of the relaxed or tensed quality of the glottis. For instance, if this slope drops sharply, this will be reflected perceptually as a relaxed voice. On the contrary, if the slope drops slowly and the glottal spectrum has a lot of treble, this will result in a rather aggressive voice and therefore perceived as a tensed voice. An estimate of the LF model [27, 31] is used to extract Rd and σ_g parameters and the vocal tract filter (VTF) is estimated by taking the estimate of

the glottal source into account. The VTF parameters are thus independent of the excitation parameters and the glottal source may be changed, keeping the VTF untouched which can be of interest for voice transformations in expressive speech synthesis. The speech model is the following: the signal is assumed to be stationary in a short analysis window (≈ 3 periods in voiced parts, $5ms$ in unvoiced parts). In the frequency domain, the voice production model of an observed speech spectrum $S(\omega)$ is (see Fig. 1):

$$S(\omega) = (H^{f_0}(\omega) \cdot G^{Rd}(\omega) + N^{\sigma_g}(\omega)) \cdot C^{\bar{c}}(\omega) \cdot L(\omega) \quad (1)$$

H^{f_0} is a harmonic structure with fundamental frequency f_0 . G^{Rd} is the deterministic excitation, i.e. an LF model [30]. This model is defined by: the fundamental period $1/f_0$, 3 shape parameters and the gain of the excitation E_e . To simplify the LF control, the parameter space is limited to a meaningful curve and a position defined by the value of Rd [30]. N^{σ_g} is a white Gaussian noise with standard deviation σ_g . C is the response of the VTF, a minimum-phase filter parametrized by cepstral coefficients \bar{c} on a mel scale. To avoid a dependency between the gains E_e and σ_g on one hand and the VTF mean amplitude on the other hand, a constraint is necessary. $G^{Rd}(\omega)$ is normalized by $G^{Rd}(0)$ and E_e is therefore unnecessary. Finally, L is the lips radiation. This filter is assumed to be the time derivative ($L(\omega) = j\omega$). The estimation method for each of the parameters can be found in [27] This method and in

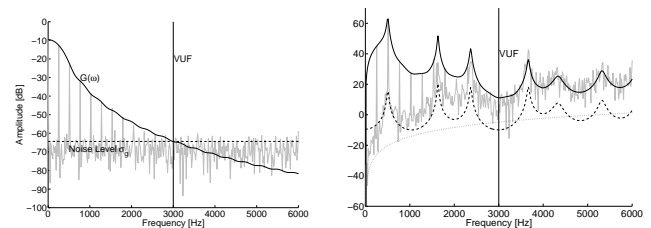


Figure 1: The mixed excitation model on the left: G^{Rd} (solid line) and the noise level σ_g (dashed line). The speech model on the right: the VTF (dashed line); L (dotted line); the spectrum of one speech period (solid line). Both plots show in gray the source $(H \cdot G + N)$ or the speech spectrum $S(\omega)$ respectively.

particular the estimation of the Rd parameter but also the *Glottal Closure Instant* (GCI) detection algorithm which is deduced from this glottal model [27, 32], allow an intuitive control of the voice quality which is of great interest for expressive speech transformations or to quickly synthesize different speaker personalities with various voice qualities from the same voice [33]. Recently Rd estimation and GCI detection have been implemented into SUPERVP which provides a wide range of voice transformations as we will see in the following.

2.3. Basic signal analysis and transformations

We now describe some of the analysis and the transformations which will be used in the following of the paper. First, the *fundamental frequency* f_0 is an important component in the transformations, as is a robust decision of whether the signal is voiced or not [34]. Another important property of the speaking voice is the fact that the harmonics in voiced segments of the signal are masked by noise above a certain frequency, which may vary from below f_0 to half of the sampling rate depending of the voice and the phonatory

setting applied. This *voiced/unvoiced frequency* will be denoted by VUF in the following. A robust estimation of the *spectral envelope* is obtained by the cepstrally based *true-envelope estimator* [35]. Compared to LPC-based methods, it has the advantages of not being biased for harmonic signals and that its order may be adapted automatically to the local f_0 . The fact that the true-envelope truly follows the spectral peaks of the signal, equips us with the control necessary for detailed filtering in time and frequency depending on the time-frequency characteristics of the signal. As long as the time variation is done with care to avoid audible discontinuities, the results keep a high degree of naturalness. Finally, as we described in the Section 2.2 the *Rd* parameter and the detection of GCI is an other important component in the transformations [33].

When it comes to the basic transformations, by (pitch) *transposition*, we mean changing the local f_0 of the signal by a certain factor while conserving the spectral envelope (as far as possible). The transposition of the spectral envelope is an independent parameter although both are done in the same operation. *Time-frequency filtering* has already been mentioned, and an other basic transformation is *time stretching*, which does not touch the frequency dimension. Finally, voice quality (breathy, harsh voice) can be transformed by modifying the glottal source characteristics via the *Rd* parameter, and GCI marks allows to introduce jitter (e.g. creaky voice).

3. TRANSFORMATION OF TYPE AND NATURE

Transformation of the voice of a given person (*source voice*) to the one of another person (*target voice*), referred to as speaker conversion, has been the subject of persistent efforts in many research labs, including IRCAM as we will see in Section 4. However, it is sometimes not necessary to reach a specific voice target. In this way, an alternative approach, which has been adopted by S. Farner in [36], rather than trying to attain a specific target voice, favors the quality of the modified sound by controlling the transformation conditions. He showed that it is nevertheless possible to change the identity of the source voice by changing its apparent size, gender and age, or making the voice breathy, softer, rougher or less happy, or even reducing it to a whisper. We first present in Section 3.1 what distinguishes voices according to voice physiology and phonatory settings. In Section 3.2 we present the corpora which were recorded to set up the different transformations presented in Section 3.3 which were implemented into TRAX, a transformation plugin presented in Section 3.4.

3.1. Differences between voices

Apart from variation in natural pitch range, different voices are distinguished and recognized by their *timbre* that depends on the physiology of the voice and the phonatory settings. The term timbre is often defined as the quality of a sound other than the pitch, duration, and loudness. For the voice we often use the term voice quality for grouping timbre-related qualities like dark, bright, soft, rich, noisy, pure, rough, etc.

3.1.1. Voice physiology

The specific configuration of the voice organ, such as the length and shape of the vocal tract and the vocal folds, varies from person to person and gives them their individual pitch range and timbre. Nevertheless, there are general differences depending on the

gender and age of the person [37, 38, 39], although it might be difficult in some cases to guess the gender and age merely from the person's voice. The most important differences are the natural vibration frequency range of the vocal folds (perceived as pitch and measured as f_0), the spectral distribution of the glottal source (for instance measured as spectral tilt), and the shape of the vocal tract (specific resonances and anti-resonances called formants and anti-formants).

Iseli et al. [39] have reported pitch means and ranges for male and female voices of ages ranging from 8 to 39 years: about 250 Hz for boys, decreasing to about 125 Hz from the age of 11 to 15 years, and about 270 Hz for girls, descending to about 230 Hz for adult women. Similar values were already published by Peterson and Barney [37] but without distinguishing boys and girls or specifying the age. However, they included average frequencies for the three first formants F1, F2, and F3 of men, women, and children for ten English vowels [37]. Averaging their formant frequencies over all vowels, we find that F1 increases about 14% from a child voice to a woman's voice, and about 33% to a man's voice. The increase is maybe slightly higher for F2, and about 18% and 38% for F3.

Finally, the aged voice presents a new set of characteristics: decreased intensity, breathiness, relatively high pitch (especially for men), lower flexibility, and perhaps trembling [40].

3.1.2. Phonatory settings

The voice can take many different phonatory settings (in addition to those necessary for making the phones of a language). For example, the vocal tract may be shaped to make a dark or bright color or sound nasal. Also interesting are the possibilities of the larynx, which has a great repertoire of voice qualities. Based on numerous studies by several researchers, J. Laver has made a comprehensive discussion and summary of the phonatory settings of the larynx and their relation to the perceived voice quality [41]. He argues for the existence of six basic phonatory settings: *modal voice* and *falsest* (orthogonal mechanisms), *whisper* and *creak* (combinable with each other and with the first category), and *breathy* and *harsh* voice. Although these are phonatory settings, such vocal qualities may also be provoked by the physical state of the voice such as fatigue, injury, and illness. J. Esling et al. have later specified the contribution to phonation of the false vocal folds and of the constrictor muscles further above, as summarized in [42]. This helps explaining constricted and unconstricted phonation modes and characterizes harsh, creaky and whispery voices as constricted phonation modes and modal voice, falsetto and breathy voice as unconstricted ones.

3.2. Recording of voice qualities

In addition to considerations of the physiology and the acoustics of the voice, one male and one female actor were recorded saying 10 sentences (in French) while faking different voice qualities depending to their abilities. The voice qualities included soft, tense, breathy, hoarse, whispering, nasal and lisping voices, as well as the voice of an old person, a child, a drunk or the effect of a stuffed nose.

Comparison with their normal voice, which was also recorded (at 48 kHz and 24 bits), gave important spectral information for the transformations, as discussed below.

3.3. Voice Transformations of type and nature

Transformations of type and nature of the voice were grouped into three categories: transformation of physical characteristics of the speaker (size, gender and age), transformation of voice quality (modification of the glottal source to make the voice breathy, whispering, rough, soft, tense, loud etc.), and transformation of speech style (modification of the prosody; liveliness, speech rate, etc.).

3.3.1. Transformation of size, gender and age

While there are general differences between voices of different gender and age, there are also considerable differences within each category. This makes it difficult to determine absolute parameters for successful transformation of the voice, and even though the parameters would be correct, the perception of gender or age may be disturbed by the fact that the speech style does not correspond to the voice. Nevertheless, with the pitch values given in Section 3.1.1 as reference, modification of pitch to change gender and age may simply be achieved by a transposition of the source signal to the given target pitch.

But merely increasing the pitch of a man's voice does only make a man speak in falsetto, or as Mickey Mouse if the spectral envelope is transposing together with f_0 and the harmonics, for instance. The vocal tract should be modified independently by transposing the spectral envelope according to an average of the ratios of the formants of men, women and children given in Section 3.1.1. In order to achieve other voices, such as a teenaged boy or girl, intermediate values were chosen.

In an interactive system, such as TRAX, a transformation plugin presented in Section 3.4, the operator can in addition be given the possibility to optimize the parameters for each voice. In some cases it may indeed be interesting to play with the ambiguity of the gender of the voice and thus choose an intermediate setting, as we did with Céladon's voice when he disguises himself as a woman in the film "Les amours d'Astrée et de Céladon" by E. Rohmer, 2007.

When it comes to aged voices, the characteristics mentioned in Section 3.1.2 are converted into transformations. A convincing old voice can be achieved by the following four actions: trembling is implemented as a slowly fluctuating transposition factor, the pitch is slightly raised (together with the spectral envelope to give a brighter timbre), the speech rate is slowed down, and the f_0 ambitus is decreased. Additionally, breathiness may be added, as described below.

Finally, no literature was found on transformation of size, but we can simply extrapolate our knowledge about gender and age transformation. The approach is intuitive, as when we read a book aloud for a child: raising the pitch and making the vocal tract smaller (transposing the spectral envelope upwards in frequency) make us sound like a small dwarf, and speaking with a low pitch and making the mouth cavity large (downwards spectral-envelope transposition) simulate the voice of a giant, for instance. Adding breathiness may be an efficient addition to a deep dragon's voice, for instance.

3.3.2. Whisper

When we whisper, the vocal folds are separated enough not to vibrate but are still sufficiently close to produce audible turbulence. Recordings showed that the spectral envelope of whisper and voiced speech are comparable at high frequencies (above the

estimated VUF) but differ at low frequencies for voiced phones. While the formant frequencies have approximately the same positions, the spectral tilt was flat or even positive below the VUF.

To transform voiced speech to whisper, a source of white noise was therefore filtered by the spectral envelope estimated from the original signal, except for some modification at low frequencies: Firstly, the spectral tilt was neutralized and even inverted below about 3 kHz, depending of the voice. The choice of 3 kHz was an empiric compromise because using the VUF as cut-off frequency for the inversion tended to create audible discontinuities. Secondly, fricatives (unvoiced phones such as /f/, /s/, /θ/, /ʃ/) should not be touched by this transformation as they depend on turbulence created at constriction further downstream. Since these noisy sounds have the energy concentrated at higher frequencies (in the range 3-6 kHz depending on the sound [24]), preserving the fricatives was indirectly achieved by the measure described above by allowing only to increase the low-frequency spectral tilt.

3.3.3. Breathy voice

A breathy phonation is obtained by reducing the force with which the vocal folds are pressed together. The vocal folds vibrate, but the closing movement is not complete or sufficiently soft for air leakage to cause turbulence noise in addition to harmonics. The effect of this on the spectrum is an increasing spectral tilt of the harmonic parts of the signal (i.e., an attenuation of high-frequency harmonics) accompanied by an addition of aspiration noise above about 2 kHz [43].

To render a voice breathy, we proceed in 2 steps: first, the voice must be softened by passing it through a lowpass filter, then noise must be added. Of course, the noise must change with the signal, which is exactly the case for the spectral envelope. An approach similar to that of whisper is thus followed, and the noise is attenuated at low frequencies to avoid it to interfere with the original signal. However, just as with whisper, the fricatives should not be touched. It is therefore important to modulate the lowpass filter with the voicing coefficient. The original, lowpass-filtered signal is then mixed with the whisperlike noise at a ratio that depends on the desired degree of breathiness.

As we have stated at the end of Section 2.2 *Rd* estimation have been recently implemented into SUPERVP so that it is now also possible to change the vocal quality (e.g. whisper, breathy) by modifying the *Rd* parameter.

3.3.4. Transformation of speech style

Differences between gender and age are also seen in terms of speech style. Speech style is much more difficult to address from a global point of view because it requires a prosodic analysis and processing. It was shown, however, that the dynamic range of the pitch, the pitch ambitus, is a speech-style attribute which varies from speaker to speaker and seems generally greater for children and teenagers than for adults, and even smaller for aged people. Changing the ambitus was efficiently achieved by exaggerating or attenuating the natural variations of f_0 by dynamically transposing the signal in proportion to the \log - f_0 deviation from the established median f_0 . The median f_0 was chosen rather than the mean f_0 because the median is invariant of the method used for this transformation. Another speech-style attribute is the speech rate. Slowing down the speech to get an aged person, for instance, was done by dilating the signal by some 20 to 50% without changing the pitch or spectral envelope. Changing the ambitus and the

speech rate together has surprising effects: dullness may well be achieved from a neutral recording by decreasing the speech rate and the ambitus. Conversely, the opposite transformation of the same neutral recording gives the effect of eagerness.

3.4. ircamTools TRAX

The study presented in the Section 3.3 led to VOICEFORGER, a library dedicated to voice transformations of type and nature based on SUPERVP. In collaboration with the company FLUX which designed the graphical interface, IRCAM, recently proposed the ircamTools commercial plug-in TRAX based on VOICEFORGER. Most of the transformations of TRAX features the interactive design of sound transformations using either real time sound input or sound files loaded into the application. Through the use of an intuitive interface, the effect of all parameter modifications applied to sound transformations can be heard in real time. TRAX is a



Figure 2: The ircamTools TRAX graphical interface by Flux (See [11] for online demos).

tool designed for voice but also music sonic transformations allowing independent transposition of pitch and timbre (spectral envelope). It offers precise control over all transformations (transposition, transposition jitter, component remixing, filtering and generalized cross synthesis). It features a creative set of presets. All parameter settings can be saved and recalled.

For single voice sounds there exist presets that allow for high quality transformations of the gender and age of the speaker. These presets can be fine tuned to the specific characteristics of the input voice. For musical sounds an additional mode for transient detection and preservation is available. When transient detection is enabled, the component remixing object allows for the independent remixing of the sinusoid, noise and transient components. Finally, transformations can be stored either as user defined presets or as SUPERVP command lines. Using command lines enables the possibility to apply batch mode transformation to many sound files at once using the settings that have been designed with TRAX.

4. SPEAKER CONVERSION

When the desired target voice is specific, it is possible to use voice conversion techniques. The aim of *Speaker Conversion* - a typical application of *voice conversion* technique (VC) - is to modify the speech signal of a source speaker to be perceived as if it had been uttered by a target speaker [44]. It can be of interest on a performative perspective. For instance, it could be used to exchange voice of speakers or singers or to convert an actor's voice towards

the voice of an disappeared celebrity or towards the voice of a famous singer. The overall methodology for speaker conversion is to define and learn a mapping function of acoustic features of a source speaker to those of a target speaker. Several approaches have been proposed such as vector quantization [45], neural networks [46] or multivariate linear regression [47] among others statistical methods. One of the most popular statistical method, proposed by Stylianou and al. [48], is based on a *Gaussian Mixture Model* (GMM) that defines a continuous mapping between the features of source and target voices.

Research on speaker conversion have been initiated at IRCAM in [49]. We recently proposed in [50] a new method for spectral conversion called *Dynamic Model Selection* (DMS) based on the use of several models in parallel, assuming that the best model may change over time according to the source acoustic features. We first recall the GMM-based spectral conversion method. Then, we introduce the DMS method in Section 4.2.

4.1. GMM-based spectral conversion

Stylianou and al. [48] proposed to model the source speaker acoustic probability space with a GMM. The cross-covariance of the target speaker with source speaker and the mean of the target speaker were then estimated using least squares optimization of an over-determined set of linear equations. Kain extended Stylianou's work by modeling directly the joint probability density of the source and target speaker's acoustic space [51]. This joint probability density is estimated on a parallel corpus in which source speaker utterance and target speaker utterance have been temporally aligned. This method allows the system to capture all the existing correlations between the source and target speaker's acoustic features. The conversion is finally performed at each frame n on the basis of the minimum mean-square error (MMSE) : the converted feature vector \hat{y}_n is the weighted sum of the conditional mean vectors $E_{k,n}^y$ in which the weights are the posterior probabilities $p(u_n = k|x_n; \phi_k)$ of the source acoustic feature vector belonging to each one of the mixture components ($u_n = k$):

$$\hat{y}_n = E[y_n|x_n] = \sum_{k=1}^K p(u_n = k|x_n; \phi_k) E_{k,n}^y \quad (2)$$

where x_n is the source feature vector at the frame n , u_n the index of the mixture component at the frame n and ϕ_k is the GMM parameters set which consists of the weight $p(u_n = k)$,

the mean vector $\bar{\mu}_k^z = \begin{bmatrix} \bar{\mu}_k^x \\ \bar{\mu}_k^y \end{bmatrix}$ and the covariance matrix $\Sigma_k^z =$

$\begin{bmatrix} \Sigma_k^{xx} & \Sigma_k^{xy} \\ \Sigma_k^{yx} & \Sigma_k^{yy} \end{bmatrix}$ for all mixture components k . The conditional covariance matrix can also be evaluated, giving a kind of confidence measure for the conditional mean vector for each $n \in \mathcal{N}$ which can be use to renormalize the variance of the converted speech parameters. Although this type of method is relatively efficient, conversion performance are still insufficient regarding speech quality: the frame by frame conversion process induces inappropriate spectral parameter trajectories and the converted spectrum can be excessively smoothed. So improvements are still necessary to make it usable for instance for artistic applications which are demanding considering quality.

4.2. Dynamic model selection

In classical speaker conversion methods, a single model is selected during the training step and used for the conversion. This model is selected among others according to the spectral distortion obtained from the conversion of a development corpus or by using methods from the models selection research field. Information Criteria such as BIC [52] have been designed for this purpose. A good model will balance goodness of fit and complexity, so it should have neither a very low bias nor a very low variance. A model with a too large variance due to overparametrization will give poor performance on data different or far from the training data because of the high variance of the local estimators resulting in overfitting. The model undergoes oscillations that are both very large and whose features strongly depend on the exact positions of the points leading to a model with a huge variance and very large response errors. However, the same model will give excellent conversion performances on datas similar or close to the training ones. The *Dynamic Model Selection* (DMS) method proposed in [50] consists of using several models in parallel assuming that the best model may change over time according to the source acoustic features. To do so, a set of potential best models \mathcal{M} including GMMs with increasing number of components is built during the training step. However, the increase of the number of components of a Gaussian mixture is limited by the increasing complexity of the model due to the large number of parameters associated with the covariance matrices. One way to solve this problem is to use diagonal structures, but the performances are then sacrificed because the latter are unable to model the underlying second order statistics.

Mixture of *Probabilistic Principal Component Analyzers* (PPCAs) is a method proposed by Tipping and Bishop [53] to solve the inflexibility of GMMs by performing a pseudo-local *Principal Component Analysis* (PCA) on each mixture component. Modeling covariance structure with a mixture of PPCAs provides an entire range of covariance structures that incrementally includes more covariance information. Mixture of PPCAs can be seen as a more general case of the GMMs for spectral conversion. It can be used in order to define models with increasing number of mixtures while keeping a reasonable model complexity. For the DMS method, several joint models including GMMs with full covariance matrices and mixture of PPCAs are estimated. Then, a set of best potential models - denoted \mathcal{M} - is selected according to the BIC criterion (the best models being the ones with the lowest BIC values).

During the conversion step, at each frame $n \in \mathcal{N}$, the most appropriate model is chosen according to the likelihood of the source data given each model as

$$\hat{M}_n = \arg \max_{M \in \mathcal{M}} p(x_n | M) \quad (3)$$

with

$$p(x_n | M) = \sum_{k=1}^K p(u_n = k) \mathcal{N}(x_n; \bar{\mu}_k^x, \Sigma_k^{xx}) \quad (4)$$

the values of $p(u_n)$, $\bar{\mu}_k^x$, Σ_k^{xx} and K , depending on the model M . In this way, we aim to use a general model with low complexity if the values are far from training data and a more complex and precise model if the source data are closer to training data, leading to a better conversion. An example of model selection along a segment of a speech utterance is given on Figure 3: complex models are used on stable spectrum parts while simpler and general models are used in transition parts.

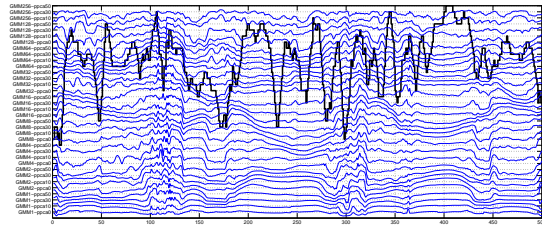


Figure 3: Example of Dynamic Model Selection along a segment of a speech utterance. On the left the set of potential models. In bold line: selected models at each frame n , in light lines: LSF representation of the source spectral envelope.

Subjective tests presented in [50] showed that the method is promising as it can improve the conversion in terms of proximity to the target and quality compared to the method based on a single model. In further work, we will focus on other criteria than the likelihood for the selection of the best model during the conversion. Finally, speaker conversion has been recently implemented into SUPERVP allowing realtime speaker conversion.

5. TRANSFORMATION OF EXPRESSIVITY

We finish this short review of voice transformations at IRCAM by presenting latest research which have been done on the transformation of expressivity in speech. This research has been initiated at IRCAM in [55]. We proposed and designed a system of transformation based on Bayesian networks trained on expressive corporuses. This system has yielded interesting results of transformation and its design was accompanied by the development of several descriptors including those concerning the degree of articulation for the expressivity. The study has been continued, which led to a second system that we briefly describe in this review. The general objective is to convert the prosody of a neutral speech into an expressive one. The proposed system, does not require any transcription or phonetic alignment of the input speech since it relies only on the acoustical data. In Section 5.1, we present the expressive corporuses and introduce the prosodic model in 5.2. We then describe the training procedure of the transformation associated to a given expressivity change in the prosodic's parameter space in Section 5.3. The last part details the generation of prosodic trajectories and the implementation of the prosodic transformations. We conclude by giving some perspectives for further work.

5.1. Expressive corporuses

We first describe the expressive corporuses that have been recorded for the training of the models of transformation. Two actors - one man and one woman - have recorded 100 utterances. These utterances were of different lengths with different number of syllables, different size of prosodic phrases and different number of breaks in order to get a broad coverage of various prosodic structures. In this work, only the first 4 basic emotions described by P. Ekman [56] were considered: *joy*, *fear*, *anger* and *sadness*. Thus, different corporuses were recorded with different intensities of emotions. The corporuses were segmented automatically by IRCAMALIGN[57], a automatic French speech segmentation software. This alignment was only used to establish a matching between neutral and expressive prosodic trajectories during the training. On the other hand,

an annotation of the prominences was performed on the neutral corpus.

5.2. Prosodic model

Prosodic modeling is achieved at the syllable level. To do so an automatic syllable segmentation has to be performed. The algorithm is introduced in the next Section 5.2.1. In Section 5.2.2, we present the different prosodic descriptors.

5.2.1. Syllable segmentation

A preliminary step of the prosodic modeling is to segment the speech into syllables. This segmentation is performed automatically from the acoustic data. The syllable detection is based on the Mermelstein algorithm [58]. The detection principle is to identify the energy dips that correspond to the boundaries of syllabic nuclei as illustrated on Fig. 4. However, the detection score was greatly

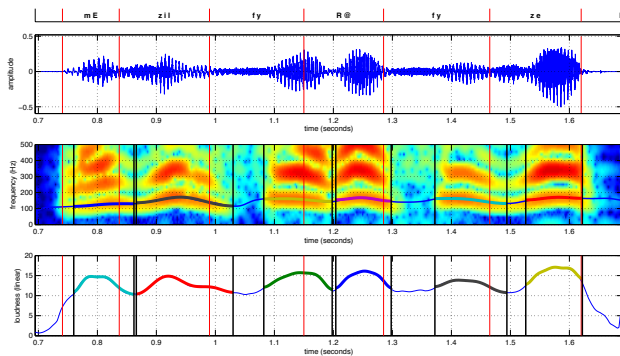


Figure 4: Detection of syllable nuclei

improved by deriving the syllable loudness from the True envelope spectrum [25].

5.2.2. Prosodic descriptors

We followed the 5-dimensional model of prosody proposed by Pfitzinger in [59], and extract the following descriptors on a syllable basis:

- **Intonation** characterizes the fundamental frequency curve. It is calculated from the syllable segmentation. Discrete Cosinus Transform (DCT) is used with a high order (7 coefficients) across the syllable. The first three factors are used to model the main speech gestures (mean, slope, curvature). The higher order coefficients are used primarily for modeling the effects of vibrato. We can then analyze the corpuses and characterized them in terms of these descriptors. A Principal Component Analysis (PCA) was made on DCT coefficients on the prominent syllables which are likely to reveal significant prosodic gestures. One can thus infer a dictionary of prototypical forms of the first principal component according to the different emotions as presented in the Figure 5. Registry information provides information on the intra-or extroverted nature of the involved emotion.
- **Speech rate** is based on the measure of the syllable rate and on the ratio between different durations of speech elements,

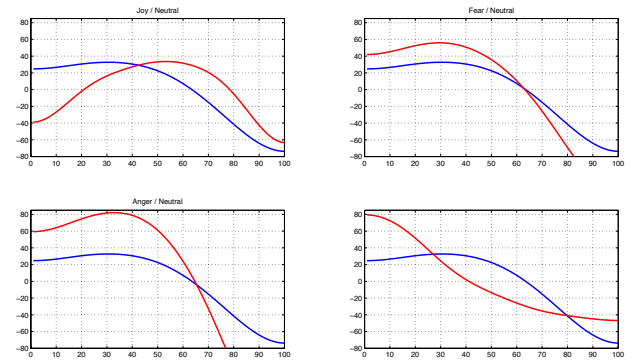


Figure 5: First principal component on DCT computed on f_0 of prominent syllables according to different emotions: top left: joy, top right: fear, bottom left: anger, bottom right: sadness. In blue line: neutral speech, in red line: expressive speech

e.g. between the duration of voiced part and unvoiced part, or between duration of active speech compared to breaks.

- **Voice quality** is mainly based on the relaxation index Rd introduced in Section 2.2. Rd measures the relaxed or tensed quality of the glottis to which we added estimates of jitter, shimmer and harmonic to noise ratio (Voiced/Unvoiced Frequency VUF). We also use the algorithm for detecting GCI also presented in Section 2.2. These marks indicate the times of closure of the glottis and allow to compute a measure of jitter. A fast evaluation shows that the neutral and sadness have a small amount of jitter compared to joy and anger.
- **Articulation** is described by estimating the dilatation coefficient between the frequency spectrum of neutral speech compared to the one of expressive speech. This coefficient is estimated by minimizing the log-spectral distance between the original and the dilated spectrums. The value of this coefficient is fixed arbitrarily for the moment and will be automatically estimated in the future.

5.3. Learning transformation functions

The principle of the system is to apply a relative transformation of the prosodic parameters for each syllable. This relative transformation depends on the context of each syllable. Since the corpuses are aligned, it is possible to define transformation vectors between each syllable of the neutral and expressive speech. In the current implementation, two separate transformation vectors are estimated:

- A transformation vector in the joint space of f_0 DCT coefficient and duration
- A transformation vector in the joint space of Rd and jitter parameters.

A context-dependent clustering of these transformation vectors is the performed according to the following contextual features:

- Position of the syllable in the utterance, categorized in beginning, middle, end of utterance
- Position of the syllable in the prosodic group (segment of the speech between 2 breaks)

- Prominent or non-prominent syllable, its kind of prominence, whether it is due to the fact that the syllable is longer than its neighbours or if it is because it has a f_0 peak higher than that of its neighbors.

A *decision tree* learning is used to build vector classes as homogeneous as possible depending on context. The training is done by minimizing the distance between the transformation vectors. Finally, the decision tree will allow to find which transformation vector to apply to a syllable according to its context, even if the context has not been observed in the training corpuses.

5.4. Transformation of expressivity

During the transformation step, vectors to use are selected according to the context of each syllable using the decision tree. To synthesize the trajectories of f_0 , the same principle used in HMM-based speech synthesis is used. Dynamic constraints (first derivative and second derivative) are taken into account during the estimation of first DCT coefficients that best explains observation data (maximum likelihood estimation). This can be written as a system of nonlinear equations which can be solved using weighted least squares. This allows to generate a smooth trajectory and finally a sort of gesture on the entire utterance. The same principle is used to generate the trajectory of f_0 , Rd and the vowels durations. Jitter and warping are modeled independently as additional effects.

During the transformation of a neutral speech, the speech signal is first segmented into prosodic groups (using voice activity detection) and syllables (using our syllable segmentation algorithm described in 5.2.1). Prominences are then automatically determined using a duration test. Then a f_0 stylization step is done by calculating the DCT coefficients for f_0 on each syllable. The decision tree is then used to determine which transformation to apply according to the context of each syllable. The different curves that are generated are applied using SUPERVP via VOICE-FORGER. Among these are transposition curves, dilatation curves for the speech rate, modification curves for Rd which is now integrated into SuperVP and jitter which is simulated by short-term transpositions.

Results depend heavily on the speech to transform. It will be necessary to learn more general transformation models in order to allow several possible strategies of transformation. A subjective evaluation of the system needs also to be done. In the longer term it would be interesting to vary the expressivity along the utterance and therefore not to have discrete categories such as it has been considered until now, which could be achieved by representing the emotions along activation and valence axis.

6. CONCLUSION

Methods for transformation of gender and age, voice qualities whisper and breathy, and speech style have been presented. With the commercial plug-in TRAX it is now possible to design and apply voice transformations in real-time using an interactive interface. Combined with TTS synthesis, it provides a powerful creation tool which can be used as a performative TTS system. When the voice target is specific, dynamic model selection can be used for speaker conversion. It will soon be implemented into SUPERVP to allow real-time speaker conversion. Finally we presented recent research on the transformation of expressivity which will surely be of interest on a performative perspective in the years to come.

7. REFERENCES

- [1] X. Rodet, "Time-Domain Formant-Wave-Function Synthesis", In *Spoken Language Generation and Understanding*, J.C. Simon, ed., D. Reidel Pub. Co., Dordrecht, Holland, pp. 429-441.
- [2] X. Rodet and P. Depalle, "Synthesis by Rule: LPC Diphones and Calculation of Formant Trajectories", In *Proc. ICASSP'85*, Tampa, FL., march 1985.
- [3] X. Rodet, P. Depalle and G. Poiriot, "Diphone Sound Synthesis based on Spectral Envelopes and Harmonic/Noise Excitation Functions", In *Proc. 1988 Int. Computer Music Conf.*, Koln, Germany, Sept. 1988, pp. 313-321.
- [4] Bennett G. and X. Rodet, Synthesis of the Singing Voice, In *Current Directions in Computer Music Research*, M. V. Mathews and J. R. Pierce, eds., The MIT Press, 1989.
- [5] A. Roebel and X. Rodet, "Efficient Spectral Envelope Estimation and its application to pitch shifting and envelope preservation", In *Proc. International Conference on Digital Audio Effects*, Madrid, 2005.
- [6] A. Roebel, "Estimation of partial parameters for non stationary sinusoids", In *Proc. International Computer Music Conference (ICMC)*, New Orleans, 2006.
- [7] A. Roebel, "A shape-invariant phase vocoder for speech transformation", In *Proc. DAFx-10*, Graz, Austria, 2010.
- [8] A. Roebel, "Shape-invariant speech transformation with the phase vocoder", In *Proc. Interspeech 2010*, Makuhari, Japan, 2010.
- [9] X. Rodet and A. Lefevre, "The Diphone program: New features, new synthesis methods and experience of musical use", In *Proc. 1997 Int. Computer Music Conf.*, 1997.
- [10] G. Peeters and X. Rodet, "SINOLA : A New Method for Analysis/Synthesis using Spectrum Distorsion, Phase and Re-assigned Spectrum", In *Proc. International Computer Music Conference*, Pekin, China, Octobre 1999.
- [11] <http://www.ircamtools.com>
- [12] G. Beller, C. Veaux, G. Degottex, N. Obin, P. Lanchantin and X. Rodet, "IRCAM Corpus Tools: Système de Gestion de Corpus de Parole", In *Proc. TAL*, 2008.
- [13] C. Veaux, G. Beller and X. Rodet, "IrcamCorpusTools: an extensible platform for speech corpora exploitation", In *Proc. LREC*, Marrakech, 2008.
- [14] C. Veaux, P. Lanchantin and X. Rodet, "Joint Prosodic and Segmental Unit Selection for Expressive Speech Synthesis", In *Proc. 7th Speech Synthesis Workshop (SSW7)*, Kyoto, Japan, 2010.
- [15] P. Lanchantin, G. Degottex and X. Rodet, "A HMM-based speech synthesis system using a new glottal source and vocal tract separation method", In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010
- [16] N. Obin, P. Lanchantin, M. Avanzi, A. Lacheret and X. Rodet, "Toward Improved HMM-based Speech Synthesis Using High-Level Syntactical Feature", In *Proc. Speech Prosody*, Chicago, USA, 2010.
- [17] N. Obin, X. Rodet and A. Lacheret, "HMM-based Prosodic Structure Model Using Rich Linguistic Context", In *Proc. Interspeech*, Makuhari, Japan, 2010.
- [18] X. Rodet, Y. Potard and J-B. Barrière, "Chant. De la synthèse de la voix chantée à la synthèse en général", In *Rapport de recherche N° 35*. IRCAM. 1985.
- [19] N. Schnell, G. Peeters S. Lemouton, P. Manoury and X. Rodet, "Synthesizing a choir in real-time using Pitch-Synchronous Overlap Add (PSOLA)", In *Proc. ICMC*, Berlin, 2000

- [20] J. Harvey, "Mortuos Plango, Vivos Voco" a realization at IRCAM, Computer Music Journal 5, MIT Press, 1981
- [21] E. Moulines and J. Laroche, "Non-parametric techniques for pitch-scale and time-scale modification of speech, In *Speech Communications*, vol. 16, pp. 175-205, 1995
- [22] Y. Stylianou, "Applying the harmonic plus noise model in concatenative speech synthesis", In *Transaction SAP*, 9(1) pp.21-29, 2001.
- [23] H. Kawahara, I Masuda-Katsuse, and A. de Cheveigne, "Re-structuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds", In *Speech Communication*, 27(3-4):187-207, 1999.
- [24] J. Laroche, "Frequency-domain techniques for high-quality voice modification", In *Proc. Int. Conf. on Digital Audio Effects (DAFx)03*, London, UK, 2003.
- [25] A. Roebel, "A new approach to transient processing in the phase vocoder", In *Proc. DAFX03*, London, UK, pp344-349, 2003.
- [26] N. Bogaards and A. Roebel, "An interface for analysis-driven sound processing", In *Proc. 119th AES Convention*, oct 2005.
- [27] G. Degottex, "Glottal source and vocal-tract separation", *Phd thesis UPMC-Ircam*, 2011.
- [28] M. Dolson, "The phase vocoder: A tutorial", In *Computer Music Journal*, vol 10, no.4, pp.14-27, 1986.
- [29] M. F. Schwartz, "Identification of speaker sex from isolated, voiceless fricatives", In *Journal of the Acoustical Society of America*, vol.43, no. 5, pp. 1178-1179, 1968.
- [30] G. Fant, "The LF-model revisited. transformations and frequency domain analysis", *STL-QPSR*, vol. 36, no. 2-3, pp. 119-156, 1995.
- [31] G. Degottex, A. Roebel and X. Rodet, "Phase minimization for glottal model estimation", *IEEE Transactions on Acoustics, Speech and Language Processing*, accepted on August 2010.
- [32] G. Degottex, A. Roebel and X. Rodet, "Glottal Closure Instant detection from a glottal shape estimate", In *Proc. 13th International Conference on Speech and Computer*, SPECOM, pages 226-231, 2009.
- [33] G. Degottex, A. Roebel and X. Rodet, "Pitch transposition and breathiness modification using a glottal source model and its adapted vocal-tract filter", In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011
- [34] A. de Cheveigné and J. Kawahara, "Yin, a fundamental frequency estimator for Speech and Music", In *Journal of the Acoustical Society of America*, vol. 111, no.4, pp. 1917-1930, 2002
- [35] A. Roebel, F. Villavicencio and X. Rodet, "On cepstral and all-pole based spectral envelope modeling with unknown model order", In *Pattern Recognition Letters*, vol. 28, pp. 1343-1350, 2007.
- [36] S. Farner, A. Roebel and X. Rodet, "Natural transformation of type and nature of the voice for extending vocal repertoire in high-fidelity applications", In *Proc. AES 35th International Conference*, London, UK, 2009.
- [37] G. E. Peterson and H. L. Barney, "Control methods used in a study of the vowels", In *Journal of the Acoustical Society of America*, vol. 24, no. 2, pp. 175-184, 1952.
- [38] K. Wu and D.G. Childers, "Gender recognition from speech. Part I: Coarse analysis", In *Journal of the Acoustical Society of America*, vol. 90, no. 4, pp. 1828-1840, 1991.
- [39] M. Iseli, Y. Shue, and A. Alwan, "Age, sex, and vowel dependencies of acoustic measures related to the voice source", In *Journal of the Acoustical Society of America*, vol. 121, no. 4, pp. 2283-2295, 2007.
- [40] R. J. Baken, "The aged voice: A new hypothesis", In *Journal of Voice*, vol. 19, no. 3, pp.317-325, 2005.
- [41] J. Laver, "The phonetic description of voice quality", *Cambridge studies in linguistics* Cambridge University Press, 1980.
- [42] L. D. Bettany, "Range exploration of phonation and pitch in the first six months of life", Master of arts, University of Victoria, 2002.
- [43] D. H. Klatt and L. C. Klatt, "Analysis, synthesis, and perception of voice quality variations among female and male talkers", In *Journal of the Acoustical Society of America*, vol. 87, no. 2, pp. 820-857, 1990
- [44] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara, "Voice conversion through vector quantization", In *J. Acoust. Soc. Jpn. (E)*, vol. 11, no.2, pp. 71-76, 1990.
- [45] M. Abe, S. Nakamura, and H. Kawabara, "Voice conversion through vector quantization", In *Proc. International Conference on Acoustics Speech and Signal Processing (ICASSP'88)*, pp. 655-658, 1988.
- [46] M. Narendranath, H. Murthy, S. Rajendran, and B. Yegnanarayan, "Transformation of formants for voice conversion using artificial neural networks", In *Speech Communication*, vol. 16, pp. 207-216, 1995.
- [47] H. Valbret, E. Moulines, and J. Tubach, "Voice transformation using psola technique", In *Speech Communication*, no 11, pp.175-187, 1992.
- [48] Y. Stylianou, O. Cappé, and E. Moulines, "Continuous probabilistic transform for voice conversion", In *IEEE Transactions on Speech and Audio Processing* vol. 6, pp. 131-142, 1998.
- [49] F. Villavicencio, A. Roebel and X. Rodet, "Improving LPC spectral envelope extraction of voiced speech by True-Envelope estimation", In *Proc. ICASSP*, Toulouse, 2006.
- [50] P. Lanchantin and X. Rodet, "Dynamic Model Selection for Spectral Voice Conversion", In *Proc. Interspeech 2010*, Makuhari, Japan, Sept 2010.
- [51] A. Kain and M. W. Macon, "Spectral voice conversion for text-to-speech synthesis", In *Proc. International Conference on Acoustics Speech and Signal Processing (ICASSP88)*, pp. 285-288, 1998.
- [52] Y.M. Bishop, S.E.Fienberg and P.W. Holland, "Discrete Multivariate Analysis", In *MIT Press Cambridge*, 1975.
- [53] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyser", In *Neural Computation*, vol. 11, no. 2, pp. 443-482, 1999.
- [54] T. Toda, A. W. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory", In *IEEE Trans. Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2222-2235, 2007.
- [55] G. Beller, "Expresso: Transformation of Expressivity in Speech", In *Proc. Speech Prosody*, Chicago, 2010.
- [56] P. Ekman, "The handbook of cognition and emotion", Basic Emotions chapter, John Wiley & Sons, Ltd., 1999.
- [57] P. Lanchantin, A. C. Morris X. Rodet and C. Veaux, "Automatic Phoneme Segmentation With Relaxed Textual Constraints", In *Proc. LREC'08 Proceedings*, Marrakech, Morocco, 2008.
- [58] P. Mermelstein, "Automatic segmentation of speech into syllabic units", In *Journal of the Acoustical Society of America*, vol. 58, pp. 880-883, 1975.
- [59] H. R. Pfiztinger, "Five dimensions of prosody: intensity, intonation, timing, voice quality, and degree of reduction", In *Proc. Speech Prosody*, paper KN2, 2006.

EFFICIENT POLYNOMIAL IMPLEMENTATION OF THE EMS VCS3 FILTER MODEL

Stefano Zambon

Dipartimento di Informatica
Università di Verona
Italy

stefano.zambon@univr.it

Federico Fontana

Dipartimento di Matematica e Informatica
Università di Udine
Italy

federico.fontana@uniud.it

ABSTRACT

A previously existing nonlinear differential equation system modeling the EMS VCS3 voltage controlled filter is reformulated here in polynomial form, avoiding the expensive computation of transcendental functions imposed by the original model. The new system is discretized by means of an implicit numerical scheme, and solved using Newton-Raphson iterations. While maintaining instantaneous controllability, the algorithm is both significantly faster and more accurate than the previous filter-based solution. A real time version of the model has been implemented under the Pure-Data audio processing environment and as a VST plugin.

1. INTRODUCTION

Within the *virtual analog* research field, several efforts have been made to properly simulate the voltage-controlled filters (shortly, VCF) onboard the monophonic synthesizers of the 60's, such as Robert Moog's transistor-based VCF [1, 2, 3], or the diode-based VCF designed for the Electronic Music System *Voltage Controlled for Studio with 3 Oscillators*, known as VCS3, which is considered in this paper.

The first discrete time model of the EMS VCS3 VCF was presented in 2008 [4]. In that model the analog filter network was accurately represented through a nonlinear differential equation system, that was later discretized by means of an explicit scheme using a fourth-order Runge-Kutta method. A similar system was reposed in 2010 [5], where a passive digital filter network directly coming out from the analog structure was computed using fixed-point iterations. This computation was proven to be efficient enough to run in real-time meanwhile allowing variation at sample rate of the VCF control parameters, typically the cutoff frequency and feedback gain.

In this paper, an evolved simulation of the previous system [5] is proposed. Specifically, the system equations are reformulated in order to avoid the expensive computation of transcendental functions, hence obtaining a quasi-polynomial system which is then discretized using an implicit scheme and Newton-Raphson iterations. Overall, the speed improvement is of an order of magnitude. Moreover, due to the improved numerical behavior, the simulation computes accurate solutions for large values of the control parameters, i.e. where the fixed-point method failed to converge in reasonable time causing noticeable artifacts in the output.

By considering a specific VCF analog circuitry, obviously this study has not the generality of recently proposed techniques for the simulation of generic electrical networks [6, 7]. However, some of the employed recipes like the removal of transcendental functions and the specific implicit scheme design can be applied to

other nonlinear systems, that need to be accurately simulated in real time.

This poster is organized as follows. In Sec. 2, the nonlinear differential state-space representation of the VCS3 VCF is shortly reviewed. Then, in Sec. 3, some algebraic manipulations are carried out to obtain an equivalent description containing polynomial functions, which substitute the hyperbolic tangent used in the original formulation. The resulting system is discretized with an implicit method that is proposed in Sec. 4, and whose implementation details are discussed. Finally, Sec. 5 shows some results that prove the improved performance of the proposed solution compared to the previous model.

2. MODEL

The VCF is a parametric filter, whose cutoff frequency and resonant behavior can be controlled respectively by varying the characteristic value of the nonlinear resistive components and the feedback gain. The behavior of the original circuitry can be described with a good approximation by the following differential equations system [5]:

$$\begin{cases} \dot{v}_{C1} = \frac{I_0}{2C} \left(\tanh \frac{v_{IN} - v_{OUT}}{2V_T} + \tanh \frac{v_{C2} - v_{C1}}{2\gamma} \right) \\ \dot{v}_{C2} = \frac{I_0}{2C} \left(\tanh \frac{v_{C3} - v_{C2}}{2\gamma} - \tanh \frac{v_{C2} - v_{C1}}{2\gamma} \right) \\ \dot{v}_{C3} = \frac{I_0}{2C} \left(\tanh \frac{v_{C4} - v_{C3}}{2\gamma} - \tanh \frac{v_{C3} - v_{C2}}{2\gamma} \right) \\ \dot{v}_{C4} = \frac{I_0}{2C} \left(-\tanh \frac{v_{C4}}{6\gamma} - \tanh \frac{v_{C4} - v_{C3}}{2\gamma} \right) \\ v_{OUT} = (K + 1/2) v_{C4} \end{cases} \quad (1)$$

In this standard space-state representation, v_{IN} and v_{OUT} are respectively the voltage input and output signals; K is the feedback gain (ranging between 0 and 10 in the VCS3 synthesizer) and I_0 is a bias current setting the resistance values, hence the cutoff frequency of the filter. The state variable vector $\mathbf{v}_C = [v_{C1}, \dots, v_{C4}]$ corresponds to the voltages through the four capacitors present in the electrical network. The other terms in Eq. (1) are the constants $\eta = 1.836$, $V_T = 26$ mV, $\gamma = \eta V_T = 48$ mV, and $C = 0.1$ μ F. The system has a fixed point at the origin, corresponding to null charge at the capacitors [5].

3. NONLINEAR SYSTEM REFORMULATION

The main complexity in the model expressed by Eq. (1) is that any accurate system solution requires the computation of several hyperbolic tangents. This computation, especially on modern hard-

ware, can be several order of magnitudes more expensive than multiplying. Thus, it would be beneficial to rewrite equations containing only polynomial functions.

In order to do so¹, we exploit the self-similarity of the derivative of the hyperbolic tangent: $d/dt \tanh(t) = 1 - \tanh^2(t)$. Then, we proceed by assigning the values of the nonlinear terms in (1) to the new auxiliary state vector $\mathbf{x} = [x_1, \dots, x_5]$:

$$\begin{cases} x_1 = \tanh \frac{v_{C_2} - v_{C_1}}{2\gamma} \\ x_2 = \tanh \frac{v_{C_3} - v_{C_2}}{2\gamma} \\ x_3 = \tanh \frac{v_{C_4} - v_{C_3}}{2\gamma} \\ x_4 = \tanh \frac{-(K+1/2)v_{C_4}}{2V_T} \\ x_5 = \tanh \frac{v_{C_4}}{6\gamma} \end{cases} \quad (2)$$

The only auxiliary variable which does not capture a portion of (1) directly is x_4 . Even if it is possible to set this variable to one argument of the hyperbolic tangent, doing so would require to include (hence to numerically compute) the derivative of the input signal v_{IN} . Since assumptions on the smoothness of the input signal cannot be made, it is preferable not to include incoming signals' derivatives into the system, as they are sensitive to noise and prone to amplification of the high frequencies.

Rather, we use the addition formula for the hyperbolic tangent and rewrite the term as

$$\tanh \frac{v_{IN} - v_{OUT}}{2V_T} = \frac{\tilde{v} + x_4}{1 + \tilde{v}x_4},$$

where $\tilde{v} = \tanh(v_{IN}/2V_T)$. In this way we are still left with one transcendent function in the system: since it includes only the signal v_{IN} , its (possibly parallel) computation across the input buffer can be decoupled by the solution of the system.

The system nonlinearities cause a bandwidth expansion on the signal, so that oversampling is necessary to avoid aliasing [5]. A good compromise between aliasing reduction and computational cost is the use of 8x upsampling, as shown in Fig. 1.

Taking the time-derivative of the new state vector \mathbf{x} , we obtain the following equations:

$$\begin{cases} \dot{x}_1 = \frac{\dot{v}_{C_2} - \dot{v}_{C_1}}{2\gamma} (1 - x_1^2) \\ \dot{x}_2 = \frac{\dot{v}_{C_3} - \dot{v}_{C_2}}{2\gamma} (1 - x_2^2) \\ \dot{x}_3 = \frac{\dot{v}_{C_4} - \dot{v}_{C_3}}{2\gamma} (1 - x_3^2) \\ \dot{x}_4 = \frac{(K+1/2)\dot{v}_{C_4}}{2V_T} (1 - x_4^2) \\ \dot{x}_5 = \frac{\dot{v}_{C_4}}{6\gamma} (1 - x_5^2) \end{cases} \quad (3)$$

Finally, substituting into (3) the expressions for the previous state

¹This transformation is generally applicable whenever the nonlinearities are compositions in their own of exponential functions.

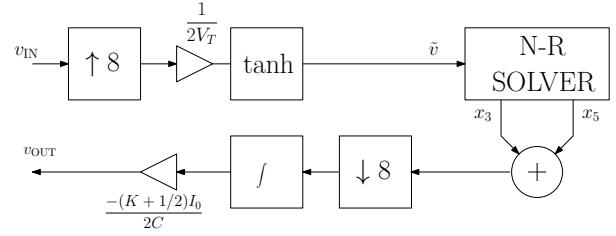


Figure 1: Block diagram illustrating the computational stages required before and after the nonlinear system solver: upsampling, nonlinear map, system solving, downsampling and integration.

\mathbf{v}_C yields the following quasi-polynomial nonlinear ODE system:

$$\begin{cases} \dot{x}_1 = \frac{I_0}{4C\gamma} \left(x_2 - \frac{\tilde{v} - x_4}{1 - \tilde{v}x_4} \right) (1 - x_1^2) \\ \dot{x}_2 = \frac{I_0}{4C\gamma} (x_3 - 2x_2 + x_1) (1 - x_2^2) \\ \dot{x}_3 = \frac{I_0}{4C\gamma} (-x_5 - 2x_3 - x_2) (1 - x_3^2) \\ \dot{x}_4 = \frac{I_0(K+1/2)}{4CV_T} (-x_5 - x_3) (1 - x_4^2) \\ \dot{x}_5 = \frac{I_0}{12C\gamma} (-x_5 - x_3) (1 - x_5^2) \end{cases} \quad (4)$$

In this system all functions are polynomial, except for a divide in the first equation due to the need to avoid the derivative of the input signal.

We can recover v_{OUT} from the relation $\dot{v}_{C_4} = (-I_0/2C)(x_5 + x_3)$. Since in (1) v_{OUT} is proportional to v_{C_4} , numerical integration (see Fig. 1) is required as a final inexpensive step to compute the solution, furthermore preserving the *passivity* of the continuous integrator if the trapezoidal rule is used for the discretization [5], as we will do through Eq. (6).

4. NUMERICAL SOLUTION

We discretize (4) using standard techniques from numerical analysis [8]. The system can be written in vectorial form as

$$\dot{\mathbf{x}} = f(\tilde{v}, \mathbf{x}), \quad (5)$$

where f is a nonlinear vectorial function. Time discretization is performed with the Adams-Moulton 1-step method (i.e., the trapezoidal rule):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{T}{2} [f(\tilde{v}_n, \mathbf{x}_n) + f(\tilde{v}_{n+1}, \mathbf{x}_{n+1})] \quad (6)$$

which is numerically equivalent to the bilinear transformation [5]. The resulting numerical equation is implicit, since at each step we need to solve a nonlinear system of equations. More precisely, we have to find the zeroes of the vectorial function

$$\begin{aligned} F(\xi) &= \mathbf{x}_n + \frac{T}{2} [f(\tilde{v}, \mathbf{x}_n) + f(\tilde{v}, \xi)] - \xi \\ &= \mathbf{F}_{0,n} + \frac{T}{2} f(\tilde{v}, \xi) - \xi. \end{aligned} \quad (7)$$

The term $\mathbf{F}_{0,n}$ has been highlighted in the expression so that at each time step it can be updated efficiently with the recursive relation

$$\mathbf{F}_{0,n+1} = 2\mathbf{x}_{n+1} - \mathbf{F}_{0,n}. \quad (8)$$

The system (7) is solved using Newton-Raphson iterations, which guarantee second-order convergence in our case. At every time step we start with the initial guess $\xi_0 = \mathbf{x}_n$, then we update iteratively the linear solution

$$\begin{aligned} JF(\xi_i) \delta_{\xi_i} &= -F(\xi_i) \\ \xi_{i+1} &= \xi_i + \delta_{\xi_i}, \end{aligned} \quad (9)$$

where $JF(\xi_i)$ is the Jacobian of F at the i -th iteration, furthermore related to the Jacobian of f by the relation $JF = \frac{T}{2} Jf - \mathbb{I}$.

Convergence is checked against the L_∞ norm of the residual vector δ_{ξ_i} using a threshold of 10^{-8} , accurate enough for single-precision floating point computations. In [5] a different condition was employed, based only on the output value. Conversely, checking all the values of the state vector can provide more accurate results especially during the simulation of transients.

The computation of the term F and the Jacobian JF can be simplified if we split the terms of the system (4) into a vector of coefficients

$$\mathbf{c} = \begin{bmatrix} I_0/(4C\gamma) \\ I_0/(4C\gamma) \\ I_0/(4C\gamma) \\ I_0(K+1/2)/(4CV_T) \\ I_0/(12C\gamma) \end{bmatrix},$$

plus two vectors, respectively containing the differences and the quadratic terms in (4):

$$\mathbf{t} = \begin{bmatrix} x_2 - \frac{\tilde{v}-x_4}{1-\tilde{v}x_4} \\ x_3 - 2x_2 + x_1 \\ -x_5 - 2x_3 - x_2 \\ -x_5 - x_3 \\ -x_5 - x_3 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 1 - x_1^2 \\ 1 - x_2^2 \\ 1 - x_3^2 \\ 1 - x_4^2 \\ 1 - x_5^2 \end{bmatrix}.$$

For sake of compactness, we have dropped the discrete-time index n in the previous equations. By (9), F can be computed from (7) using the obvious relation

$$f(\tilde{v}, \mathbf{x}) = \mathbf{c} * \mathbf{t} * \mathbf{d}, \quad (10)$$

while the Jacobian of f is written in terms of the new vectors as

$$Jf = \begin{bmatrix} 2c_1(d_1+t_1x_1) & -c_1d_1 & 0 & c_1d_1f_v & 0 \\ -c_2d_2 & 2c_2(d_2+t_2x_2) & -c_2d_2 & 0 & 0 \\ 0 & -c_3d_3 & 2c_3(d_3+t_3x_3) & 0 & c_3d_3 \\ 0 & 0 & c_4d_4 & 2c_4t_4x_4 & c_4d_4 \\ 0 & 0 & c_5d_5 & 0 & c_5(d_5+2t_5x_5) \end{bmatrix} \quad (11)$$

where $f_v = (\tilde{v}^2 - 1)/(1 - \tilde{v}x_4)^2$.

Note that the linear system described by this Jacobian matrix does not have any particular structure. For this reason, we have to employ general linear solvers such as LU decomposition with pivoting or QR factorization. In our tests, LU with pivoting was slightly less accurate, occasionally requiring an extra iteration to converge, but generally 25% faster than QR decomposition.

5. DISCUSSION

The model has been implemented both as an offline Matlab simulation, as a C++ *external* running under the PureData [9] real-time audio processing environment and as a VST [10] Plugin. For the real-time versions, we have employed the library *libresample* [11]

for accurate upsampling and downsampling, and the *Eigen* processing library [12] for linear system solving and parallelized vector computations. The real-time model requires less than 10% CPU power on an Intel Core2 Duo@2.4Ghz laptop, independently of the VCF parameter values.

The main advantage of the new algorithm resides in significantly faster, and parameter-independent computation times compared to the previous reference model [5]. We can give a rough quantitative comparison considering the approximate number of MPOS (multiplication per input sample) required by either implementation. At every step the previous algorithm requires the computation of 5 hyperbolic tangents, plus 4 discrete-time integrations and 10 multiply-and-accumulate (MAC) operations. If we approximate the cost of each hyperbolic tangent to 50 MPOS², then the cost for each fixed-point iteration amounts to about 280 MPOS. Since the average number of iterations is between 10 to 50, we can estimate the total cost per (upsampled) time step as ranging between 3000 and 15000 MPOS. In [5] some examples of the number of iterations required for different control values are given.

As opposed to the previous procedure, the proposed algorithm instead requires to compute just one hyperbolic tangent. Since this function can be precomputed on the input buffer at 1/4 the sampling frequency, in parallel to the system integration, the cost of this computation is around 5 MPOS. Building the right-hand term F as by (10) requires roughly 20 MPOS, including the divide, and the same cost is required for the computation of the Jacobian matrix (11). LU pivoting takes about 120 MPOS, resulting in a total cost of 165 MPOS per iteration. Finally, thanks to the better numerical behaviour of the Newton-Raphson method, only 3-6 iterations are usually required for convergence. Therefore, the total cost of the proposed algorithm is between 500 and 1000 MPOS, about one order of magnitude less than the fixed-point model.

As a by-product, the new model is more accurate for high values of the control parameters, especially the cutoff frequency. In fact, iteration upper bounds are inevitably reached by the fixed-point solver under these conditions [5]. The effect is illustrated in Fig. 2, where the cutoff frequency is linearly increased across the simulation. It can be easily noticed that the previous solver (left plots) behaves inappropriately around cutoff values amounting to 10 KHz, when the number of iterations starts to reach the upper bound set to 100 iterations. Within ranges of the control parameters that do not cause iteration explosion, both models introduce a relative error below -80 dB at every step, in ways that their output differences are inaudible.

6. CONCLUSION

We have transformed the equations proposed for the simulation of the VCS3 VCF, by deriving a quasi-polynomial system allowing an implicit integration based on Newton-Raphson iterations. Compared to the previous one, the new solution is significantly faster as well as more accurate when high values of control parameters are used, meanwhile enabling full controllability of the parameters at sample rate without artifacts.

7. ACKNOWLEDGMENTS

The authors would like to thank Marco Civolani for his inspiring comments on this work.

²Benchmarked on an Intel Core2Duo@2.4Ghz with gcc 4.6.

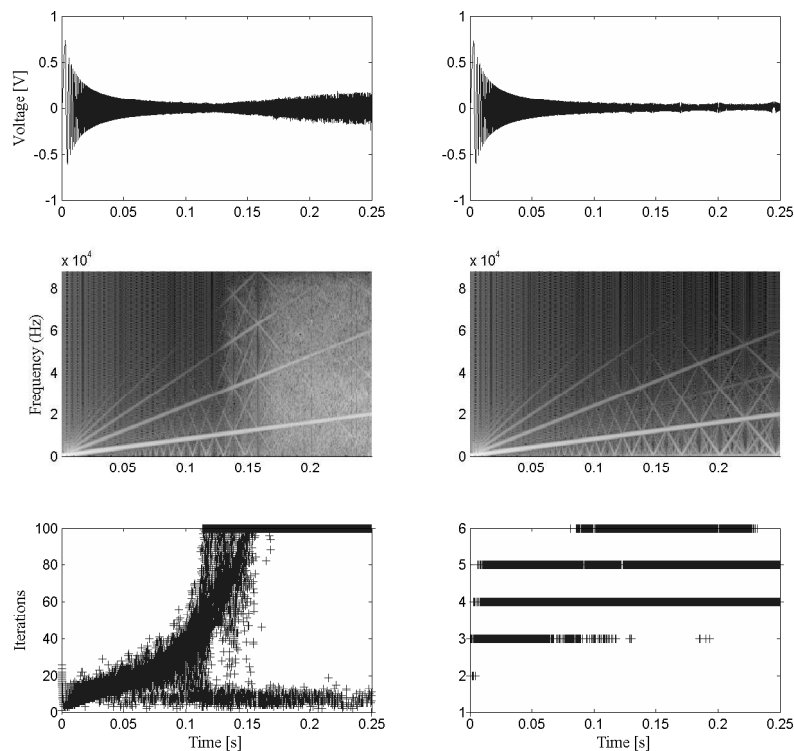


Figure 2: Comparison of the proposed model (right) versus the previous implementation (left). The input signal is a sinusoidal sweep ranging from 0 to 20 KHz, while the cutoff frequency is varied between 10 and 20KHz. The feedback gain is kept constant ($K = 1$) during the simulation. Sampling frequency set at 176400 Hz.

8. REFERENCES

- [1] T. Stilson and J.O. Smith, "Analyzing the Moog VCF with considerations for digital implementation," in *Proceedings of the International Computer Music Conference*, 1996, pp. 398–401.
- [2] A. Huovilainen, "Nonlinear digital implementation of the Moog ladder filter," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 61–64.
- [3] F. Fontana, "Preserving the structure of the Moog VCF in the digital domain," in *Proc. Int. Comput. Music Conf.*, Copenhagen, Denmark, 2007, pp. 27–31.
- [4] M. Civolani and F. Fontana, "A nonlinear digital model of the EMS VCS3 voltage-controlled filter," in *Proceedings of the 11th International Conference on Digital Audio Effects*, Helsinki, Finland, 2008, pp. 35–42.
- [5] F. Fontana and M. Civolani, "Modeling of the EMS VCS3 voltage-controlled filter as a nonlinear filter network," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 4, pp. 760–772, 2010.
- [6] D.T. Yeh, J.S. Abel, and J.O. Smith, "Automated Physical Modeling of Nonlinear Audio Circuits for Real-Time Audio Effects-Part I: Theoretical Development," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 4, pp. 728–737, 2010.
- [7] F. Fontana and F. Avanzini, "Computation of delay-free nonlinear digital filter networks: Application to chaotic circuits and intracellular signal transduction," *Signal Processing, IEEE Transactions on*, vol. 56, no. 10, pp. 4703–4715, 2008.
- [8] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*, Springer Verlag, 2007.
- [9] M. Puckette, "Pure Data," in *Proc. International Computer Music Conference*, Thessaloniki, Greece, 1997.
- [10] Steinberg Soft and Hardware GMBH, "Steinberg virtual studio technology (vst) plug-in specification 2.0 software development kit," 1999.
- [11] D. Mazzoni, "libresample," <http://ftp.debian.org/pool/main/libr/libresample/>.
- [12] Gael Guennebaud, Benoit Jacob, et al., "Eigen v3," <http://eigen.tuxfamily.org>, 2010.

Keynote 3 - Fourier + 200 - *Patrick Flandrin*

Exactly 200 years ago, Joseph Fourier wrote his fundamental essay on heat diffusion, introducing mathematical tools that have been and still are central in the development of signal analysis and processing. Many variations around Fourier's seminal approach have then been proposed, especially for the sake of facing time-varying and/or nonstationary situations. The purpose of this talk is to present and illustrate some recent methodological advances in such directions, from wavelet-like transforms to sparse time-frequency distributions and oscillations-based empirical mode decompositions.

Patrick Flandrin is a CNRS Senior Researcher, working in the Physics Department of ENS de Lyon, France. His research interests are mostly in nonstationary signal processing (time-frequency/time-scale methods), scaling processes and complex systems. He published over 250 journal or conference papers in those areas, contributed several chapters to collective books and authored one book. He has been awarded the Philip Morris Scientific Prize in Mathematics (1991), the SPIE Wavelet Pioneer Award (2001), the Prix Michel Monpetit from the French Academy of Sciences (2001) and the Silver Medal from CNRS (2010). Fellow of IEEE (2002) and EURASIP (2009), he has been elected Member of the French Academy of Sciences in 2010.

INTERACTION-OPTIMIZED SOUND DATABASE REPRESENTATION

Ianis Lallemand,

UMR STMS
IRCAM–CNRS–UPMC
Paris, France

ianis.lallemand@ircam.fr

Diemo Schwarz

UMR STMS
IRCAM–CNRS–UPMC
Paris, France

diemo.schwarz@ircam.fr

ABSTRACT

Interactive navigation within geometric, feature-based database representations allows expressive musical performances and installations. Once mapped to the feature space, the user's position in a physical interaction setup (e.g. a multitouch tablet) can be used to select elements or trigger audio events. Hence physical displacements are directly connected to the evolution of sonic characteristics — a property we call *analytic sound-control correspondence*. However, automatically computed representations have a complex geometry which is unlikely to fit the interaction setup optimally. After a review of related work, we present a physical model-based algorithm that redistributes the representation within a user-defined region according to a user-defined density. The algorithm is designed to preserve the analytic sound-control correspondence property as much as possible, and uses a physical analogy between the triangulated database representation and a truss structure. After preliminary pre-uniformisation steps, internal repulsive forces help to spread points across the whole region until a target density is reached. We measure the algorithm performance relative to its ability to produce representations corresponding to user-specified features and to preserve analytic sound-control correspondence during a standard density-uniformisation task. Quantitative measures and visual evaluation outline the excellent performances of the algorithm, as well as the interest of the pre-uniformisation steps.

1. INTRODUCTION

1.1. Background

In this study, we focus on interactive musical performances and installations based on navigation within a sound database. The user selects database elements and triggers events (for instance the playback of a sample) by physical navigation in an interaction setup, which is mapped to a geometric database representation.

Sound databases may include data as diverse as full-length recordings, samples, sound grains (used for instance by corpus-based concatenative synthesis methods [1]) or even non-audio elements such as synthesizer presets. In most cases, it is possible to build feature-based data representations by automatic and quantitative measuring of each database element's sonic characteristics. They can be quantified using sound descriptors for instance [2], or the parameters of a synthesizer preset. Such representations are closely related to the field of content-based music information retrieval, which has attracted much attention in the past years as it allows greater insight on the data than usual keywords classification.

Besides simplifying the process of working with large databases, feature-based representations have one important property: they guaranty that the elements' positions in the database representation are connected to their sonic characteristics. To illustrate this property, we consider the example of the CataRT software [3], a corpus-based concatenative synthesizer which provides screen-displayed, 2D representations of sound grains databases, obtained by computing their sound descriptor values (cf. figure 1). In this example we have used Spectral Centroid¹ and Periodicity² descriptors. We use the computer mouse as an interaction setup: a simple way to control the synthesis is to trigger the playback of a grain when the point representing it on the screen is hovered by the mouse cursor. Because our representation is feature-based, small mouse movements result in playing similar-sounding samples, while larger movements allow the user to pick grains with greater sonic differences. Furthermore, a gesture along a coordinate axis results in keeping constant one characteristic of the selected grains, while controlling the remaining property (for instance playing grains of same brightness but of different harmonic characters). We use the name *analytic sound-control correspondence* to define this property.

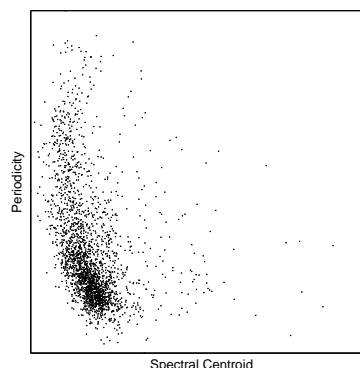


Figure 1: Database representation using Spectral Centroid and Periodicity descriptor values.

Common descriptions of sound data use a large number of features, which yield high-dimensional database representations. Such representations are usually not suited for physical navigation, which is usually performed in two or three dimensions. Hence new

¹The Spectral Centroid (measured in Hertz) is defined as the center of gravity of the FFT magnitude spectrum. It is closely related to the perception of brightness.

²The Periodicity measures the harmonic character against the noisy character of the sound.

database representations have to be used for this purpose; they can either be obtained by dimensionality reduction methods or by direct selection of the features the user wants to control.

Our study addresses a problem relative to the use of a 2D interaction setup, though it can virtually be extended to any number of dimensions. Examples of such interaction setups are XY controllers, multitouch tablets, or even the floor of an exhibition room combined with a position tracking device (in this case, selection would be performed by the user's displacements in the room).

1.2. Problem

Each interaction setup has its own fixed geometry: it would be a rectangle for a XY controller, a disk for the Reactable [4], or any possible shape for an exhibition room. It is very unlikely that the mapping between this geometry and the geometry of the 2D database representation obtained by dimensionality reduction or feature selection will be optimal (in a user-defined sense). Hence we need a flexible way to adapt the representation geometry to that of our interaction setup, while preserving as much as possible the analytic sound-control correspondence property — one of the strengths of feature-based representations.

To illustrate this problem, consider that we want to sonify an exhibition space using our previous example database and a 2D representation. Our interaction setup is the floor of the exhibition space. A camera is used to perform position tracking of the visitors, so that each visitor's physical navigation in the room create a path through the database representation. A sample is triggered each time this path intersects a sample's position. It is as if we had mapped the database representation to the surface of the room, and a sample was triggered each time a visitor's position intersected a sample's position. Suppose that we seek full sonification of the exhibition space, i.e. that we care to use all the interaction space that is available to us for controlling the samples, and that we want higher sample density near the top-right corner of the room. Figure 2 shows two sonifications achieved with different representations, which are superposed to the room blueprint.

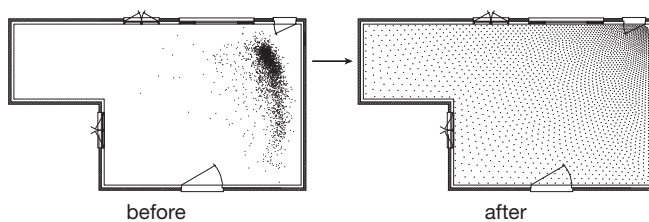


Figure 2: Examples of sonification of an exhibition space using a 2D database representation.

On the left, the representation shown in figure 1 has been used. Simple transformations such as symmetries and scaling obviously preserve analytic sound-control correspondence. Though we have been able in this case to put a high density region near the top-right corner, we obviously cannot obtain by these means a new representation that fits perfectly the boundaries of the room. In all cases a large amount of “silent” zones will be left, in which no samples can be triggered. Clearly, manually moving points so that we obtain a representation that roughly fits the room geometry when superposed to its blueprint is not a good solution to our problem, as it would be very difficult for us to ensure that the new

representation retains some aspects of the analytic sound-control correspondence property. Furthermore, moving points manually gets more and more difficult as the database size increases.

On the contrary, a geometric algorithm might take care of automatically finding a new representation that suits our geometric constraints, while preserving analytic sound-control correspondence as much as possible. The right part of figure 2 shows a database representation obtained with *unispring*, an algorithmic solution we have provided to this problem (detailed in section 3). Besides having been able to make the new representation fit the geometry of the room, we have kept a higher density region near the top-right corner, as required by the sonification design we had in mind.

1.3. Algorithmic Solution

We present in section 3 *unispring*, a physical model-based algorithm which allows to spread the original representation points within a user-defined region. The expected data point density can be specified using the mathematical framework detailed in section 3.1.

The algorithm was designed to preserve analytic sound-control correspondence as much as possible; an evaluation of its performance is presented in section 4.

2. RELATED WORK

2.1. Previous Work

We had previously addressed the problem of locally controlling the density of a database representation using mass spring damper-based algorithms. They helped avoiding overlapping points by pushing them apart, and have been used in a dimensionality-reduction algorithm [5]. Other analytic sound-control correspondence preserving algorithms were also developed [6], but proved to be less efficient than the solution detailed in this article.

2.2. Data Visualisation Methods

Dynamic visualisation methods, such as zoom and pan, allow accurate selection of individual points in high-density regions. These approaches might also adjust the number of points to display accordingly to the zoom level, in order to avoid data overlap. However, such methods apply when the user interacts with the help of a visual feedback of the database current state, which is not always the case (as shown for instance in section 1.2).

On a more fundamental level, also note that our aim was not to provide dynamic data display, but geometrically transforming the representation of the data to obtain an optimal, “static” interface for navigation. By static, we mean that this interface is determined once by the user, and doesn't require further adjustments in the course of the interaction process.

2.3. Dimensionality Reduction

Dimensionality reduction algorithms such as PCA [7], weighted PCA [8] or MDS [5] can be used to obtain 2D representations from original high-dimensional feature-based representations. Another approach is to project the representation onto a plane by selecting the two features users will be able to control. Dimensionality reduction methods might be helpful in cases where it is unclear

which features to control, or when it appears that no pair of features amongst those immediately available by data analysis will provide satisfactory results in terms of expressivity. The latter case might particularly be true for applications using sounds of very different natures, such as those contained in a collection of field recordings for instance. However, note that projection by selection of two features naturally maps their perceptual meaning to the two degrees of freedom the user interacts with. By taking care of preserving the analytic sound-control correspondence property, our algorithm also preserves this perceptual meaning.

2.4. The *distmesh* Algorithm

The *distmesh* algorithm, available as a Matlab toolbox [9], generates unstructured triangular meshes using a physical algorithm. It is based on a simple mechanical analogy between a triangular mesh and a 2D truss structure, or equivalently a structure of springs. It provides a mathematical framework that allows the user to specify the internal geometry of the mesh as well as the region over which it has to be generated. Whereas *distmesh* is aimed at generating a mesh over a blank region, the physical algorithm part of *unispring* (detailed in section 3.3) adapts the physical model used in *distmesh* to relocate previously existing points (the initial feature-based database representation).

3. THE UNISPRING ALGORITHM

The *unispring* algorithm works in two parts. It starts by performing a pre-uniformisation of the 2D initial database representation (3.2), before iteratively applying a physical model-based algorithm (3.3). The algorithm spreads the representation points within a user-defined region, inside which the local density can be specified.

3.1. User-specified Features

The user-defined region is represented by its *signed distance function*, which gives the distance from any point in the plane to the closest region boundary. This function takes negative values inside the region and equals zero on its boundary. Analytic computation of the signed distance function is possible for simple regions, such as square and circular ones. For more complex regions, the function has to be computed numerically. We provided support for polygonal regions, using an iterative method implemented in [10].

The user can specify the final data point density by providing a desired length function $h(x, y)$. If (x, y) are the coordinates of the middle of two points, $h(x, y)$ gives a target distance between these points that should be reached after applying the algorithm. The resulting distances are in fact proportional to those specified by $h(x, y)$, which actually gives the relative distance distribution over the region. Density-uniformisation can thus be obtained by providing any uniform length function.

3.2. Pre-uniformisation

The pre-uniformisation steps provide equally spaced coordinate values on each axis. Since the expected final density of data points is user-specified, the pre-uniformisation steps can be seen as a way to provide a “neutral” distribution of data points that will allow more efficient action of the subsequent physical algorithm.

- **step 1:** on each coordinate axis,

- **step 1a:** sort the coordinate value list $(x_i)_{1 \leq i \leq N}$ (resp. $(y_i)_{1 \leq i \leq N}$). For each position i , get the position $n(i)$ in the sorted list.
- **step 1b:** fill output coordinate value list $(x'_i)_{1 \leq i \leq N}$ such as $x'_i = n(i)$ (resp. $(y'_i)_{1 \leq i \leq N}$).
- **step 2:** normalize the resulting coordinate values so that all data points lie inside the user-specified region.

Figure 3 shows the intermediate database representation obtained after applying steps 1 and 2 to the representation shown in figure 1.

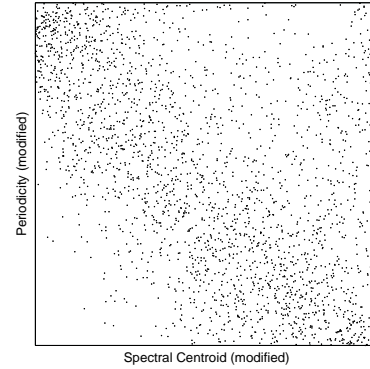


Figure 3: Database representation using pre-uniformised Spectral Centroid and Periodicity descriptor values.

3.3. Physical Algorithm

The uniform algorithm takes as input the coordinate lists $(x'_i)_{1 \leq i \leq N}$ and $(y'_i)_{1 \leq i \leq N}$ obtained by pre-uniformisation, and outputs new coordinate lists $(x''_i)_{1 \leq i \leq N}$ and $(y''_i)_{1 \leq i \leq N}$. It works according to a physical analogy: a Delaunay triangulation of the data point distribution is performed, which defines a truss structure where edges of the triangles (the connections between pairs of points) correspond to bars, and points correspond to joints of the truss.

Each bar of the truss structure has a force-displacement relationship $f(l, l_0)$ depending on its current length l and its unextended length l_0 (which is computed using the desired length function). To help points spread out across the whole user-defined region, only repulsive forces are allowed:

$$\begin{cases} f(l, l_0) = k(l_0 - l) & \text{if } l < l_0 \\ f(l, l_0) = 0 & \text{if } l \geq l_0 \end{cases} \quad (1)$$

Points that move outside the user-defined region during the algorithm iteration are moved back to the closest boundary point. This corresponds to the physical image of external reaction forces on the truss structure, that act normal to the boundary; hence points can move along the boundary, but not go outside.

The 2D truss structure obtained by triangulating the distribution bounds each point to its initial closest neighbors. Repulsive forces computed along the structure bars are not likely to create very large internal movements: hence we expect each point to keep the same neighborhood throughout the process. Whether large displacements should happen, they are handled by retriangulating the current set of points so that subsequent algorithm iterations can still rely on a valid physical analogy of the distribution

as a truss structure. With such properties, the algorithm is likely to produce representations that retain some aspects of the analytic sound–control correspondence property.

- **step 3:** perform a Delaunay triangulation of the points distribution.

while !exit

- **step a:** update data point positions.
- **step b:** move points that went outside the user-defined region to the closest boundary point.
- **step c:** if all points have moved less than a significant distance, **exit = true**.
- **step d:** if points have moved more than the maximum allowed distance in respect to the previous triangulation, perform a Delaunay triangulation of the distribution.

Figure 4 shows various distributions obtained after applying the *unispring* algorithm to the representation shown in figure 1. They provide examples of square, circular and polygonal user-defined regions with uniform point densities, as well as non-uniform user-defined point density (shown in the case of a square region).

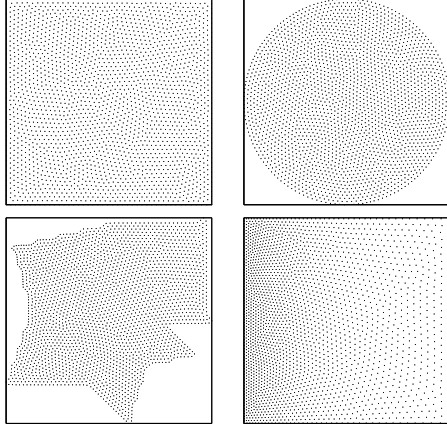


Figure 4: Database representations obtained with the *unispring* algorithm. From left to right, top to bottom: square, circular and polygonal user-defined regions with uniform point densities; non-uniform user-defined point density within a square region.

4. EVALUATION

4.1. Objectives

The aim of the evaluation process was to measure the algorithm performance relative to its expected features: redistribution of data points inside a user-defined region with internal user-defined density, and as much preservation as possible of the analytic sound–control correspondence property. Our methodology combines graphical results with quantitative measures.

Subjective visual appreciation of the algorithm action is a simple yet efficient way to make sure the final representation is constrained into the user-defined region. Evaluation was carried out using a square region.

The algorithm ability to produce a representation corresponding to the user-defined point density is evaluated in the standard case of a density-uniformisation task. The algorithm performance is assessed using a measure λ based on the normalized standard error of the 1-nearest-neighbors distance distribution [11]. For a set of N points $(P_i)_{1 \leq i \leq N}$, we define λ by

$$\lambda = \frac{1}{\bar{\gamma}} \left(\frac{1}{N} \sum_{i=1}^N (\gamma_i - \bar{\gamma})^2 \right)^{\frac{1}{2}}, \quad (2)$$

where

$$\gamma_i = \min_{j=1, \dots, N, j \neq i} P_i P_j \text{ for } i = 1, \dots, N$$

and

$$\bar{\gamma} = \frac{1}{N} \sum_{i=1}^N \gamma_i.$$

For a perfectly uniform point distribution, $\lambda = 0$; hence the smaller the value of λ , the more uniform the distribution. However, λ does not measure the algorithm performance in terms of constraining points into the user-defined region. Hence it is important to pair this measure with visual inspection of the final representation.

How the algorithm preserves analytic sound–control correspondence is estimated using three quantitative measures. To see if this property is altered, we look at pairs of points: an exception to the analytic sound–control correspondence property is introduced if at least one pair of points exchange their coordinate values on one or two axes after applying the algorithm. For instance, if points P_1 and P_2 have initial x-coordinate values such as $x_1 < x_2$, an exception is introduced if new coordinates are such that $x'_1 \geq x'_2$. The first two measures used for evaluation consist in the percentage p_1 of data point pairs that have exchanged coordinate values on one axis, and the percentage p_2 of pairs that have exchanged coordinate values on two axes. The third measure used for evaluation takes into account both p_1 and p_2 to compute a “distortion measure” value d that increases as the algorithm introduces more exceptions to the analytic sound–control correspondence property. More weight is given to pairs that have exchanged both coordinates. In our evaluation, we use

$$d = 5p_1 + 10p_2. \quad (3)$$

During the evaluation process, we refer to the pre-uniformisation steps of *unispring* (3.2) as the *uniform* algorithm, and the physical algorithm steps (3.3) as the *spring* algorithm. We compare these algorithms to the *unispring* algorithm, as they can be used independently for density uniformisation inside a square region (both were initially designed for that purpose [6]). By construction, the *uniform* algorithm fully preserves the analytic sound–control correspondence; but by doing so, we expect its performances in terms of density-uniformisation to be inferior to those of other algorithms. Evaluating the *spring* algorithm alone allows assessing the interest of the pre-uniformisation steps in *unispring*.

4.2. Evaluation Data

The evaluation was performed on five representations, providing different geometric configurations. They were obtained us-

ing CataRT sound-descriptors computation capabilities, and manual selection of two features.

- the *madeleine* representation (2404 points), with Spectral Centroid and Periodicity features, is made of subway sounds and presents a single-centered point distribution.
- the *gaussian* representation (2404 points), with 2-centered Gaussian artificial feature values on each axis, corresponds to a critical case with two very distinct centers.
- the *wave* representation (2404 points), with Start Time and Periodicity features, is made of environmental sounds and presents no identifiable distribution center.
- the *partita* representation (388 points), with Spectral Centroid and Note Number features, is made of instrumental sounds and presents initial coordinate values ranges dominated by the coordinate values of a few marginal points.
- the *alarm* representation (779 points), with Spectral Centroid and Note Number features, consists of different samples from the Freesound online library³. Points associated with the same MIDI notes appear as horizontal lines in the 2D display.

4.3. Graphical Results

Figures 5, 6, 7, 8 and 9 show the original evaluation representations and the final representations obtained by applying the *uniform*, *spring* and *unispring* algorithms. Unsurprisingly, the *uniform* algorithm alone is unable to fulfill the density-uniformisation task set by the user. The *spring* and *unispring* algorithms both seem to give satisfying results, providing square-shaped representations.

4.4. Quantitative Measures

Table 1 gives the values of the quantitative measures λ , p_1 , p_2 and d defined in section 4.1.

By construction, the *uniform* algorithm fully preserves the analytic sound-control correspondence property ($d = 0$). However, this “zero-distortion” action prevents the algorithm from efficiently performing the tasks set by the user : besides failing at producing a square-shaped representation, this algorithm is responsible for higher λ values than any other algorithm, which underlines its lower performance in terms of density-uniformisation.

In contrast, *spring* and *unispring* are associated with lower and comparable λ values. With λ values respectively equal to 3.7 % and 3.8 % of the original representations’ λ values on average, both algorithms successfully performed the density-uniformisation task. One necessary drawback to this is the introduction of distortion, though in different amounts. Since the absolute values of p_1 , p_2 and d depend strongly on the initial representation geometry, algorithms have to be compared based on their action on a single representation. Reviewing the results representation by representation we see that the *unispring* algorithm provides lower values of d , which means that it is better at preserving analytic sound-control correspondence. This can also be noted by looking at percentages p_1 and p_2 , which are lower in the case of *unispring*.

The mean values of p_1 and p_2 obtained with *unispring* are $\bar{p}_1 = 24.13$ % and $\bar{p}_2 = 0.49$ %. On average, a quarter of data point pairs have exchanged their coordinates on one axis, but the

more preoccupying case of pairs having exchanged their coordinates on two axes only happened for a very few of them. Hence the algorithm performs well at preserving analytic sound-control correspondence. This property can be partly accounted for by the physical analogy on which *unispring* is based, as explained in section 3.3. As *spring* is also based on the same physical model, greater performances of *unispring* confirm the interest of its pre-uniformisation steps. They prevent the subsequent physical algorithm from having to process high-density regions of points, which are more likely to create anarchic movements during early iterations.

Pre-uniformisation is a much faster process than the physical algorithm, which gets iterated many times: consequently, the *spring* and *unispring* algorithms’ speeds can be compared by this iteration count alone. Using this criteria, the *unispring* algorithm proved to be 1.2 times faster on average than *spring*. Taking only into account the number of retriangulations happening during the iteration process (cf. **step d** in section 3.3), *unispring* appeared to be twice faster on average than *spring*.

5. APPLICATIONS AND FUTURE WORK

5.1. Digital Musical Instrument Interfaces

Interactive corpus-based concatenative synthesis, combined with a two-dimensional single- or multi-touch control surface, can be used as a digital musical instrument (DMI) to “play” a sound database by navigation through the feature space. Here, our algorithm allows to exploit the totality of the interaction surface for control, while preserving the original analytic sound-control correspondence property and thus the perceptual meaning of the instrumental gestures.

The algorithm has been used to lay out more than 2500 sound segments for interaction on a multi-touch surface controlling the CataRT system⁴ in the piece *Alarm-Signal* performed by Diemo Schwarz at the Sound and Music Computing Conference, Barcelona 2010.⁵

A video documentation can be found in the on-line journal *Musimédiane* [12].

5.2. Preset Interpolation

Another prospective application is the layout of many synthesiser or effect presets on a 2D plane for position-based preset interpolation, where the proximity of the cursor to the position of the closest presets determines their influence on the sound [13, 14] (see also *pMix*⁶ for Max/MSP).

When a large number of presets is used, manual positioning might become unfeasable, and our automatic layout algorithm could again optimise the interaction space.

5.3. Comprehensive User Interface

The expected final point density has to be provided to the algorithm in the form of the desired length function $h(x, y)$. We studied several ways of making this process more user-friendly, with the aim of providing automatic calculation of h in the most common cases.

⁴<http://imtr.ircam.fr/imtr/CataRT>

⁵<http://mtbf.concatenative.net>

⁶<http://www.olilarkin.co.uk/index.php?p=pmix>

³<http://www.freesound.org>

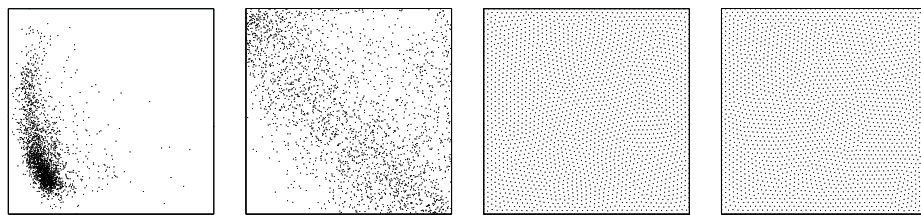


Figure 5: Madeleine representation. From left to right: original representation, uniform, spring, unispring algorithm representations.

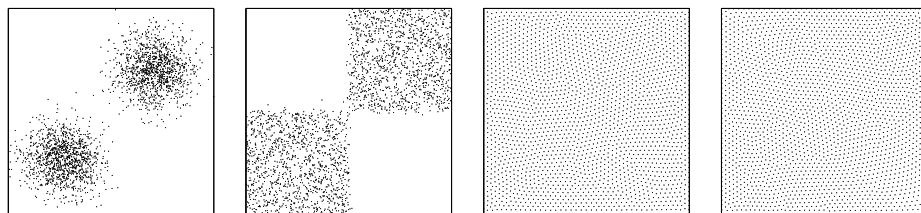


Figure 6: Gaussian representation. From left to right: original representation, uniform, spring, unispring algorithm representations.

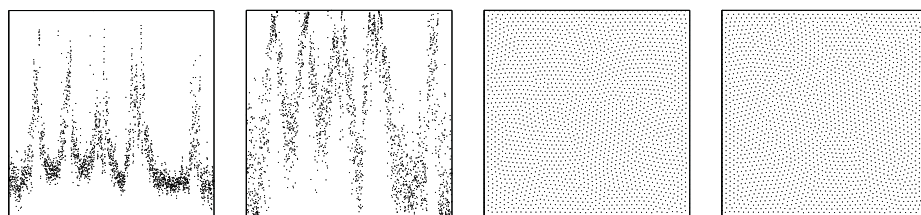


Figure 7: Wave representation. From left to right: original representation, uniform, spring, unispring algorithm representations.

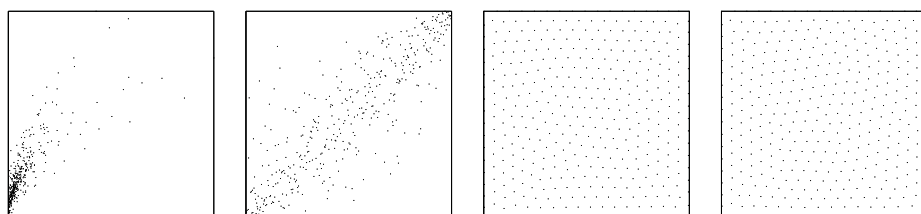


Figure 8: Partita representation. From left to right: original representation, uniform, spring, unispring algorithm representations.

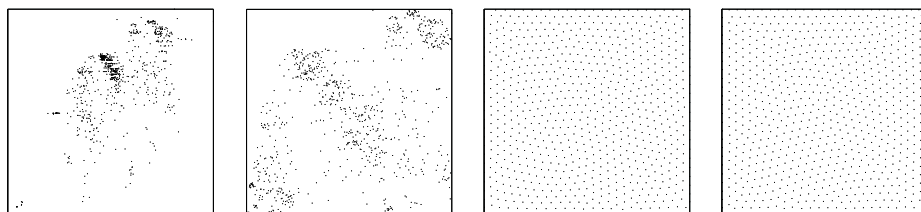


Figure 9: Alarm representation. From left to right: original representation, uniform, spring, unispring algorithm representations.

Algorithm	Measure	<i>madeleine</i>	<i>gaussian</i>	<i>wave</i>	<i>partita</i>	<i>alarm</i>
<i>none</i> (reference)	λ	1.7812	0.9876	0.9379	2.0258	1.3090
<i>uniform</i>	λ	0.6388	0.5288	0.7179	0.7994	0.8614
	p_1	0 %	0 %	0 %	0 %	0 %
	p_2	0 %	0 %	0 %	0 %	0 %
	d	0	0	0	0	0
<i>spring</i>	λ	0.0484	0.0503	0.0457	0.0518	0.0463
	p_1	41.8778 %	34.1827 %	27.4539 %	43.1718 %	28.2426 %
	p_2	9.8234 %	2.6994 %	2.4235 %	6.6142 %	2.8476 %
	d	3.0762	1.9791	1.6150	2.8200	1.6969
<i>unispring</i>	λ	0.0448	0.0487	0.0445	0.0482	0.0498
	p_1	21.6047 %	28.7839 %	16.6518 %	33.7426 %	19.8733 %
	p_2	0.54385 %	0.88407 %	0.46855 %	0.17227 %	0.36181 %
	d	1.1346	1.5276	0.8794	1.7044	1.0298

Table 1: Values of quantitative measures λ , p_1 , p_2 and d .

For instance, partially-uniform representations can be obtained by using

$$h(x, y) = \frac{1}{d(x, y) + d_0},$$

where $d(x, y)$ is the positive original representation density (computed using kernel density estimation [15] for instance) and d_0 a constant used to modulate the amount of uniformisation. When d_0 becomes much greater than $d(x, y)$ for all x, y , h tends to a uniform length function providing total uniformisation (3.1). Figure 10 shows partially-uniform representations obtained with this method.

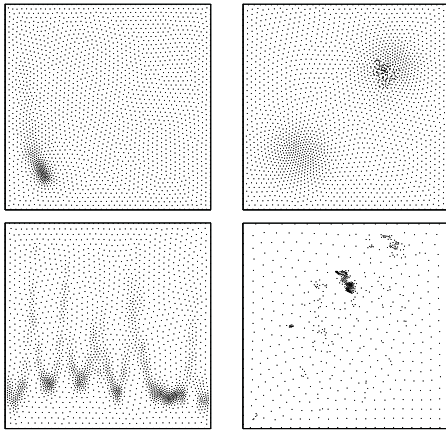


Figure 10: Partially-uniform representations obtained by automatically computing h from the initial density. Original representations used (left to right, top to bottom): *madeleine* ($d_0 = 5$), *gaussian* ($d_0 = 2$), *wave* ($d_0 = 1$), *alarm* ($d_0 = 1$).

5.4. Task-based User Evaluation

By providing a convenient, user-defined way to redistribute a database representation, our algorithm could possibly make sound design tasks easier. To test this hypothesis, we are currently considering conducting a task-based user evaluation, where users would be asked to execute basic sound-design actions by interacting with both transformed and original representations.

6. CONCLUSIONS

The *unispring* algorithm is an efficient solution to transform 2-dimensional database representations into user-specified representations. It is based on a physical algorithm whose performances are increased by pre-uniformisation of the data. The user defines a region inside which the algorithm redistributes the data points according to a user-defined length function, which determines the final distribution density.

Two-dimensional representations provide a convenient framework for interacting with sound, as in many contexts two degrees of freedom are used for interaction. Automatically extracting sound features from the database elements allows to create representations of large databases in which analytic correspondence between interaction control and evolution of sound characteristics can be found. With such representations it is often straightforward to create a mapping between the interaction setup and the sound data. However, it is unlikely that the user will consider this mapping optimal. Our algorithm allows the user to redistribute the data points in a way that will suit his needs more, while taking care of preserving as much as possible the analytic correspondence between interaction control and sound characteristics.

Our evaluation process provides information on how much this correspondence is altered by applying the algorithm. With a reasonable mean value of 24.13 % of data points pairs that have exchanged their coordinate values on one axis and only 0.49 % of pairs that have exchanged their coordinate values on two axes, the algorithm preserves most aspects of the analytic sound-control correspondence property. The algorithm shows excellent perfor-

mance when asked to perform the density-uniformisation task used for evaluation.

Unispring applications may include live performances, sound installations, sound design or educational courses. In order to make it easier for users to use the algorithm, we plan to release its real-time implementation in Max/MSP as an MnM object [16] by the time of the conference.

7. ACKNOWLEDGMENTS

The work presented here is partially funded by the *Agence Nationale de la Recherche* within the project *Topophonie*, ANR-09-CORD-022, <http://topophonie.fr>.

8. REFERENCES

- [1] D. Schwarz, “Corpus-based concatenative synthesis,” *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 92–104, 2007, Special Section: Signal Processing for Sound Synthesis.
- [2] G. Peeters, “A large set of audio features for sound description (similarity and classification) in the Cuidado project,” Tech. Rep. version 1.0, Ircam – Centre Pompidou, Paris, France, Apr. 2004.
- [3] D. Schwarz, R. Cahen, and S. Britton, “Principles and applications of interactive corpus-based concatenative synthesis,” in *Journées d’Informatique Musicale (JIM)*, GMEA, Albi, France, Mar. 2008.
- [4] S. Jordà, “On stage: the reactable and other musical tangibles go real,” *International Journal of Arts and Technology*, vol. 1, pp. 268–287, 2008.
- [5] D. Schwarz and N. Schnell, “Sound search by content-based navigation in large databases,” in *Proceedings of 6th Sound and Music Computing Conference (SMC’09)*, Porto, Portugal, Jul. 2009.
- [6] I. Lallemand, “Optimized visualisation for navigation in large sound databases,” M.S. thesis, Université Pierre et Marie Curie, Sep. 2010.
- [7] J. Shlens, “A tutorial on principal component analysis,” Dec. 2005.
- [8] D. Skočaj, A. Leonardis, and H. Bischof, “Weighted and robust learning of subspace representations,” *Pattern Recogn.*, vol. 40, no. 5, pp. 1556–1569, 2007.
- [9] P. Persson and G. Strang, “A simple mesh generator in Matlab,” *SIAM Review*, vol. 46, pp. 2004, 2004.
- [10] I. M. Mitchell, “The flexible, extensible and efficient toolbox of level set methods,” *J. Sci. Comput.*, vol. 35, no. 2-3, pp. 300–329, 2008.
- [11] H. Nguyen and J. Burkardt and M. Gunzburger and L. Ju and Y. Saka, “Constrained CVT meshes and a comparison of triangular mesh generators,” *Computational Geometry*, vol. 42, no. 1, pp. 1 – 19, 2009.
- [12] A. Bonardi, F. Rousseaux, D. Schwarz, and B. Roadley, “La collection numérique comme paradigme de synthèse/composition interactive,” *Musimédiane: Revue Audiovisuelle et Multimédia d’Analyse Musicale*, , no. 6, 2011, <http://www.musimediane.com/numero6/COLLECTIONS/>.
- [13] A. Momeni and D. Wessel, “Characterizing and controlling musical material intuitively with geometric models,” in *Proceedings of the 3rd International Conference on New Interfaces for Musical Expression (NIME’03)*, Montréal, Canada, May 2003, pp. 54–62, National University of Singapore.
- [14] A. Freed, J. MacCallum, A. Schmeder, and D. Wessel, “Visualizations and Interaction Strategies for Hybridization Interfaces,” in *Proceedings of the 10th International Conference on New Interfaces for Musical Expression (NIME’10)*, 2010, pp. 343–347.
- [15] B. W. Silverman, *Density estimation: for statistics and data analysis*, Chapman and Hall, London, 1986.
- [16] F. Bevilacqua, R. Müller, and N. Schnell, “MnM: a Max/MSP mapping toolbox,” in *Proceedings of the 5th International Conference on New Interfaces for Musical Expression (NIME’05)*, Singapore, Singapore, May 2005, pp. 85–88, National University of Singapore.

REALTIME SYSTEM FOR BACKING VOCAL HARMONIZATION

Adrian von dem Knesebeck, Sebastian Kraft and Udo Zölzer

Department of Signal Processing and Communications
Helmut Schmidt University
Hamburg, Germany
audio@hsu-hh.de

ABSTRACT

A system for the synthesis of backing vocals by pitch shifting of a lead vocal signal is presented. The harmonization of the backing vocals is based on the chords which are retrieved from an accompanying instrument. The system operates completely autonomous without the need to provide the key of the performed song. This simplifies the handling of the harmonization effect. The system is designed to have realtime capability to be used as live sound effect.

1. INTRODUCTION

The task to synthesize various voices from a solo singing voice has been realized in many different ways. Some approaches aim to synthesize a whole choir from a single singing voice. A system for the synthesis of natural sounding choir voices was presented in [1]. The singing voice is modified in pitch, time and timbre to synthesize a number of choir voices. Hence the choir voices are directly synthesized from the singing voice signal.

Another approach, presented in [2], extracts high level features from the singing voice and synthesizes the choir voices using a database of voices and timbres. The singing voice is morphed with the database voices to synthesize a choir which contains the features of the single voice, but also the smooth sound of a choir.

In this paper we present a system which synthesizes backing vocals. In contrast to the choir synthesis, where the aim is to synthesize a smooth and broad choir, we are more interested in synthesizing a number of distinguishable voices like in a typical band with one lead singer and one or more backing singers. Both of the referenced approaches implement systems which require additional information on how to harmonize the choir voices. We present a system which autonomously performs the harmonization task based on a harmony analysis of an accompanying instrument, e.g. a rhythm guitar or piano. An overview of the proposed system is given in Section 2. The signal analysis and feature extraction part, which includes the pitch detection and the chord detection, is described in Section 3. The synthesis of the backing vocals by modification of the singing voice and the harmonization is described in Section 4. Section 5 describes the evaluation of the system followed by a brief discussion of the performance. In Section 6 we conclude the paper.

2. SYSTEM DESCRIPTION

The presented system consists of three main blocks, i.e. the pitch detector, the chord detector and the voice synthesizer, as shown in Fig. 1. Two input signals are required for the processing. One input is the singing voice signal (*Voice*), which is fed to the pitch

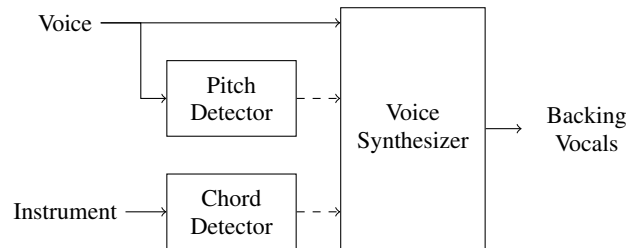


Figure 1: *System overview.*

detector for pitch extraction and to the voice synthesizer. The other input is the accompanying instrument signal (*Instrument*), which is fed to the chord detector. The singing voice pitch and the chord information are fed to the voice synthesizer. The voice synthesizer processes the singing voice input signal in respect of rules for the harmonization. The particular blocks are described in more detail in the following sections.

3. SIGNAL ANALYSIS

The signal analysis section of the system consists of a pitch detector and a chord detector. The focus of this paper is on the chord detection particularly the multipitch detection required for the chord detector.

3.1. Pitch Detection

It is important to have robust information of the current lead vocal pitch to enable the succeeding blocks to work properly. We chose the YIN algorithm as presented in [3] for the pitch detection, because it returns robust and accurate results for monophonic harmonic signals [4].

3.2. Chord Detection

The task of the chord detector is to analyze the polyphonic instrument signal to extract multiple pitches and to derive from these pitches the corresponding chord symbol representation. The extraction of multiple pitches requires to reduce the rich harmonic signal produced by a musical instrument to the pitches. For most instruments the pitches can be expressed as the fundamental frequencies $F0$.

3.2.1. Multipitch detection

The multipitch detection algorithm is based on an approach proposed by Tolonen [5]. The Tolonen system is the auditory moti-

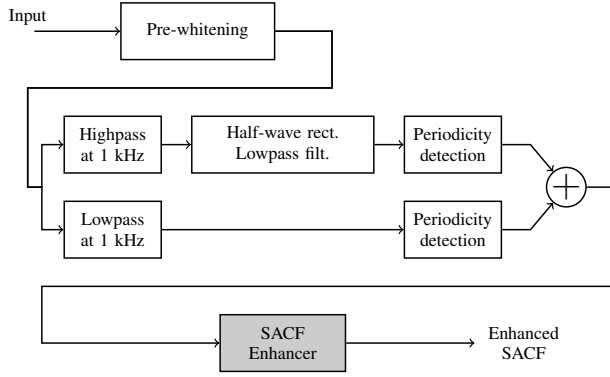


Figure 2: Tolonen's multipitch detector [5].

vated system as shown in Fig. 2. The input signal is filtered by a whitening filter and split into two bands. The periodicity of the lower frequency band is calculated as main indicator of the fundamental frequencies. The signal in the higher frequency band is half-wave rectified and lowpass filtered. This models the mechanical to neural transduction of an inner hair cell according to Meddis [6]. The output of the model has an ac and a dc component. The ac component is of the same frequency as the input signal and the dc component is a monotonic saturating function of signal level, which provides the envelope of the signal. The periodicity of the higher band emphasizes the lower band's periodicity. The periodicities are calculated using the Fourier transform, which allows magnitude compression and speeds up the computation compared to the time domain autocorrelation. The sum of both paths builds up the sum autocorrelation function (SACF).

The Tolonen system continues with the SACF Enhancer which removes redundant and spurious information by stretching the SACF and subtracting the stretched SACF from the original one.

We replace this block and come up with a different approach to remove redundancy to be able to obtain the fundamental pitch candidates. Starting from the SACF we calculate a threshold as

$$th = \frac{\alpha}{L} \sum_{l=0}^{L-1} \max(\text{SACF}(l), 0), \quad (1)$$

which is the mean of the first $L = 700$ lag values l of the SACF, where the SACF is truncated to have only positive values. The constant α lowers the threshold by scaling the truncated mean. We used a factor of $\alpha = 0.4$. The SACF values below the threshold are omitted and the lags M of the peaks above the threshold are further considered. As the next step we group the peaks into harmonically related periods and calculate a rating value K for each group as

$$K = \sum_{i=0}^4 2^i \cdot \max(\text{SACF}(m_i)), \quad (2)$$

with

$$\left\lfloor 2^i M - (0.7 \cdot 2^i M)^{\frac{1}{3}} \right\rfloor \leq m_i \leq \left\lfloor 2^i M + (0.7 \cdot 2^i M)^{\frac{1}{3}} \right\rfloor. \quad (3)$$

m_i describes an observation range to find the actual peak location near $2^i M$ in the SACF, because the multiples must not be exact integer multiples. Starting from low lag values for each peak lag M above th the group of corresponding subharmonics is considered and the weighting factor is determined according to (2). The lag of the highest peak of the group with the highest value of K is regarded as most prominent pitch period candidate. The j -multiples of M are removed from the SACF with $j \in \{1, 2, 3, 4, 6, 8\}$. The pitch period determination is an iterative process with the number of iteration steps given by the maximum number of determinable fundamental pitches.

The schedule of the algorithm to distinguish the most prominent $F0$ is as follows:

1. calculate SACF of a 4096 samples time frame at $f_s = 44.1$ kHz
2. calculate threshold th
3. find maxima above th with maximum lags M
4. for each M search for multiples, i.e. subharmonics
5. sum the weighted values of the group of harmonically related maxima to get a rating of harmonicity
6. select the highest rating value as $F0$ -candidate
7. eliminate the corresponding maxima of the $F0$ -candidate from the SACF
8. continue iterating from step 3 with the remaining peaks
9. stop if desired number of $F0$ -candidates is retrieved

Figure 3 shows an example of the removal process. In Fig. 3a the peaks of the SACF are shown. The group with the highest rating value K starts from $l = 112$. The peaks used for calculation of K are marked by circles. All corresponding lag values belonging to this group are marked by stars. The highest peak of the group is taken as the fundamental pitch candidate of this group, in this case the peak at $l = 225$. The marked peaks are removed as shown in Fig. 3b and the peaks of the second group, with lag values corresponding to $l = 178$, are marked. This procedure is repeated for the third group with lag values corresponding to $l = 74$ (see Fig. 3c). We end up with three lag values which represent the periods of the $F0$ -candidates. Figure 4a shows the SACF over time of an example G major chord, played on a guitar. In Fig. 4b the results of the $F0$ -candidate retrieval is shown. We see that this method is prone to octave errors, but mapping these candidates to the corresponding tones of the 12-tone chroma vector shows that we obtain a robust pitch class representation of the chord. In Fig. 4c the discrete chromagram of the detected tones is shown.

3.2.2. Chord classification

The multipitch detection and pitch class determination, respectively, is the fundamental component of the chord detection. Once the chord tones are detected, as described in Section 3.2.1, a mapping from multiple tones to the corresponding chord representation is done. This mapping can be a quite complex task considering the amount of possible chords which could occur. The classification of chords can be done using statistical models, e.g. hidden Markov Models (HMM), as presented in [7, 8]. A realtime implementation of the HMM classifier is possible with a modified Viterbi [9] at a high computational cost adding some latency. We require a low complexity algorithm and therefore we applied a classification known as pattern or template matching [10, 11, 12, 13], which

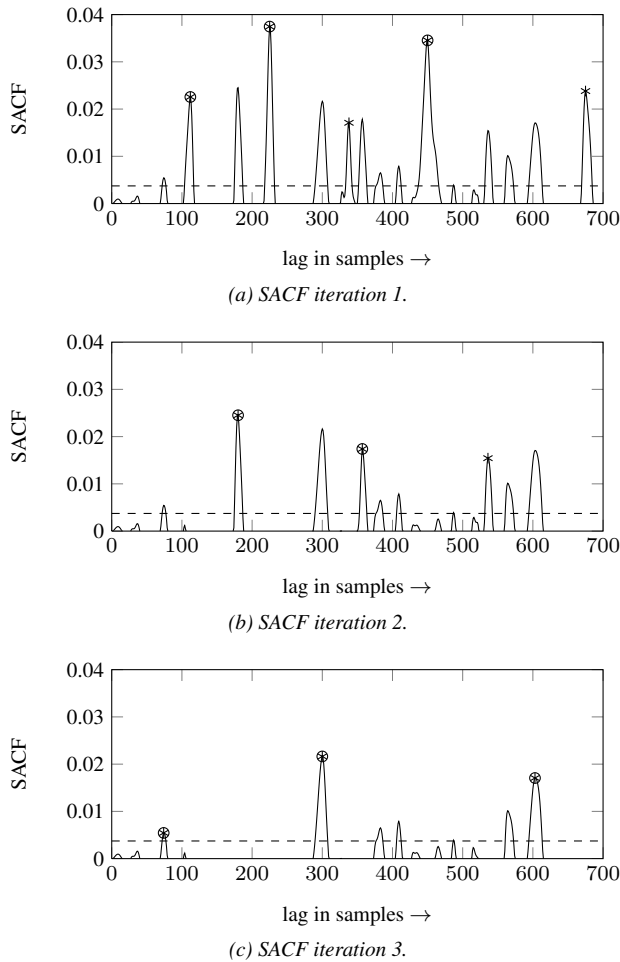


Figure 3: The three iteration steps of SACF peak removal are shown. The dashed line shows the peak detection threshold th , the stars indicate the peaks of the most prominent harmonics group and the circles indicate the peaks used for K calculation.

is applicable in a frame-by-frame manner. The detected tones are represented by a bit mask which is a 12×1 vector having a 1 where a note is present and a 0 else. With the tones ordered as $[c, c\#, d, d\#, e, f, f\#, g, g\#, a, a\#, b]$, the G major chord of the example sample would be represented as $[0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1]$. The classification is done by calculating the hamming distance between the bit mask of the detected tones with the template bit masks of the possible chords. The bit mask with the minimum hamming distance is regarded as the chord candidate. We simplify the chord detection by restraining the possible chords to two triad modes, major and minor, with the common 12 key notes per octave. This leads to a total of 24 possible chords that can be detected.

3.2.3. Validation of proposed system

The comparison of the Tolonen system with the proposed modified system was done on recorded clean guitar samples. The test set included 14 samples of standard chords (major and minor, in different keys). Each sample presents one chord which was struck once and let ring to fade out. The compared quantities include

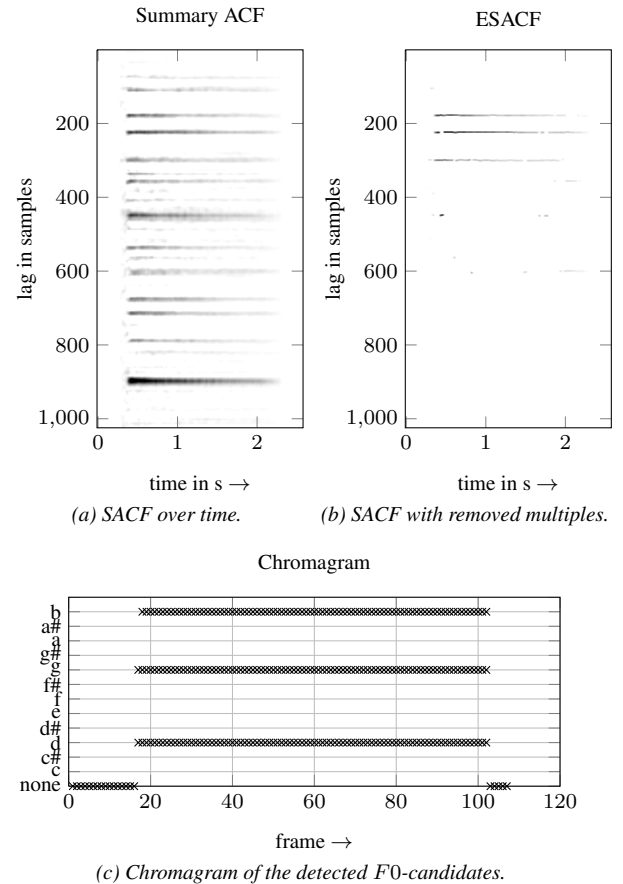


Figure 4: Results of the chromagram determination for G major guitar chord sample.

the error of the detected chords (chord error), the amount of not detected tones (tone false negative) and the amount of wrongly detected tones (tone false positive). The errors were calculated as the mean frame error. For the tone detection we were not interested in the correct octave of the played note and therefore we did not consider octave errors as false detection. The robustness of both systems was increased in the same way by temporal smoothing of the detected tones and the detected chords.

error type	Tolonen	proposed system
chord error	30.8%	2.5%
tone false negative	5.8%	1.5%
tone false positive	87.5%	12.2%

Table 1: Comparison of Tolonen approach to proposed system.

The test results show that the investigated tone detection errors could be reduced with the proposed modified Tolonen system. Consequently the chord detection error was reduced as well.

4. VOICE SYNTHESIZER

Now that we have detected the pitch F_0 of the singing voice and the chord of the accompanying instrument we can continue with

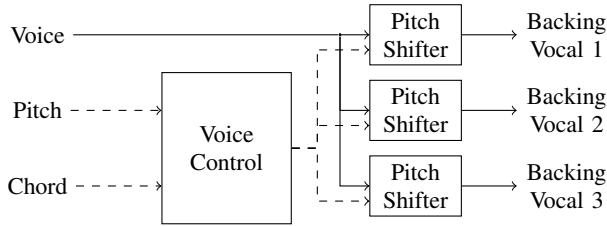


Figure 5: Background vocal synthesizer.

the synthesis of the backing vocals. The block diagram of the background vocal synthesizer is shown in Fig. 5. The synthesizer consists of a *Voice Control* block, which performs the control of the pitch shifting factors of the *Pitch Shifter* blocks. The Voice Control and the approach used for the pitch shifting are described in the following sections.

4.1. Pitch Shifter

The pitch shifters used are based on the Pitch Synchronous Overlap Add (PSOLA) pitch shifting algorithm [14, 15, 16, 17]. A block diagram of the PSOLA stages is shown in Figure 6. The al-

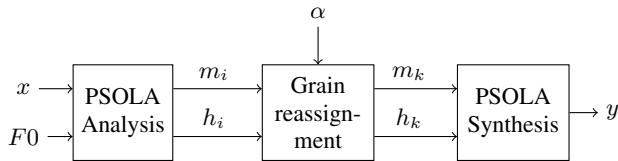


Figure 6: Block diagram of PSOLA algorithm.

gorithm segments the input signal x into short overlapping grains h_i with a length of twice the fundamental period time $1/F0$ in the analysis stage. The time instants marking the center of each grain are the analysis pitch marks m_i . Hence the time difference between two succeeding pitch marks represents one pitch period. The pitch is then shifted by a factor α in the grain reassignment stage by repositioning these grains, either reducing the distance in combination with occasional repetition of some grains to increase the pitch or expanding the distance in combination with omitting some grains to decrease the pitch. The pitch marks m_k indicate the time instants with the corresponding reassigned grains h_k for the overlap and add synthesis of the output signal y .

A robust pitch mark positioning algorithm which achieves high quality results is used as presented in [18]. The pitch mark positioning algorithm of the PSOLA analysis stage receives the fundamental pitch $F0$ from the pitch tracker described in Section 3.1. The pitch marks are positioned based on a center of energy approach, which ensures robust segmentation of the grains and reduces the occurrence of artifacts. The algorithm allows the positioning of the pitch marks in a frame-based manner to enable realtime application.

4.2. Voice Control and Harmonization

The task of the Voice Control block is to set the pitch shifting factors for the backing vocals synthesis in a way that a musically correct harmonization is achieved. We regard a harmonization as cor-

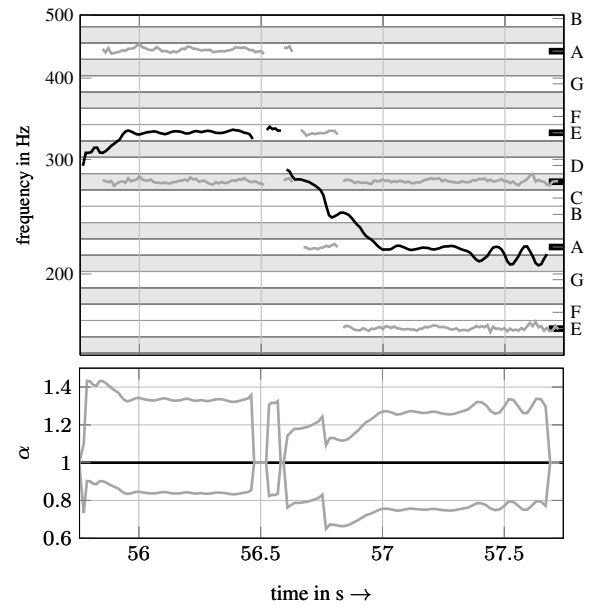


Figure 7: Harmonization example for an A major chord. The upper plot shows the pitch contours of the recorded lead vocals (black curve) and a higher and lower synthesized backing vocal (gray curves). The lower plot shows the corresponding pitch shifting factors α .

rect if the synthesized voices match the chord harmonies. At this stage of the work this means no dissonant voicings are desired.

4.2.1. Harmonization

There are numerous approaches to harmonize additional voices to the lead vocals. In offline processing the voice leading for the synthesized voices can be manually provided as score transcription. This allows the user to individually adjust every note of each voice and hence offers the most creative possibilities.

An approach towards semi-autonomous harmonization is to provide the key of the song and to define the note interval of the synthesized voice related to the singing voice. This enables to synthesize the backing vocals with the correct pitch of the notes from the scale corresponding to the provided key. The major drawback of this approach is that it still requires to manually provide additional information about the musical content.

We achieve a completely autonomous harmonization without the requirement to supply information about the musical context. The musical information is determined using the chord detector which provides the instantaneous chord information. To make the valid note determination more robust the detected chords are observed over several frames. This ensures that short failures of the chord detection do not disrupt the harmonization of the backing voices. The harmonization starts by relating the singing voice to the current chord harmony. This can be regarded as a quantization of the singing voice pitch $F0_{\text{Lead}}$ to the chord notes. We consider an example of a singer accompanied by a guitar with an A major chord being the played harmony. The upper plot of Fig. 7 shows the lead vocal pitch contour of the example as black curve. On the right hand side the chord notes for the example A major chord

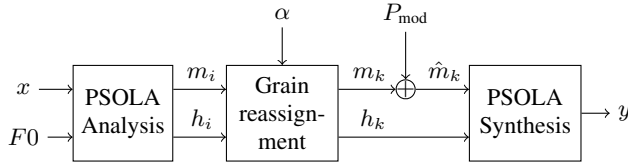


Figure 8: Block diagram of PSOLA algorithm with modulation of the synthesis pitch marks.

are marked with a black bar, i.e. A, C[#] and E. The pitch shifting factors for the backing vocals are calculated as the ratio between the singing voice pitch and the intended backing vocal pitch as

$$\alpha_{\text{Back1}} = \frac{F0_{\text{Back1}}}{F0_{\text{Lead}}}, \quad (4)$$

$$\alpha_{\text{Back2}} = \frac{F0_{\text{Back2}}}{F0_{\text{Lead}}}. \quad (5)$$

The pitch shifting range is limited to synthesize voices which are close to the singing voice, because high pitch shifting factors result in disturbing artifacts. This leads to adding a backing voice which is pitched one chord tone higher than the lead vocal pitch and a backing voice one chord tone lower. The resulting pitch shifting factors are shown in the lower plot of Fig. 7. The corresponding pitch contours of the resulting backing vocals are shown by the gray curves in the upper plot of Fig. 7.

4.2.2. Humanization

To reduce artifacts and achieve a natural sounding synthesis the pitches of the backing vocals are slightly modulated. The PSOLA algorithm allows to efficiently realize a pitch modulation for vibrato simulation and a temporal modulation for varying delay simulation using the same modulation function but with different sets of parameters. This can be accomplished by modulating the synthesis pitch mark positions as shown in Fig. 8.

The pitch mark modulation function P_{mod} is given as

$$P_{\text{mod}}(t) = \frac{A_{\text{mod}}}{2} \cdot (1 + \sin(2\pi f_{\text{mod}} \cdot t)). \quad (6)$$

A relatively high modulation frequency f_{mod} of 5-10 Hz in conjunction with a relatively low modulation depth A_{mod} in ms results in a perceivable pitch modulation. In contrast a relatively low modulation frequency of 1 Hz or lower in conjunction with a higher modulation depth results in a varying delay. The effect of the later case is shown in Fig. 9. The example shows the synthesis pitch mark modulation for $A_{\text{mod}} = 45$ ms and $f_{\text{mod}} = 1$ s. The upper plot shows the synthesis pitch marks m_k and the resulting modulated synthesis pitch marks \hat{m}_k . The lower plot shows the modulation function $P_{\text{mod}}(t)$.

5. EVALUATION AND DISCUSSION

The described harmonization system is intended to support a lead singer of a small group or a solo artist with backing vocals. The current system is applicable for the typical singer songwriter kind of musical style. The development of the particular system modules was done in Matlab. The Matlab implementation of the algorithms was already done in a realtime manner. This allowed to

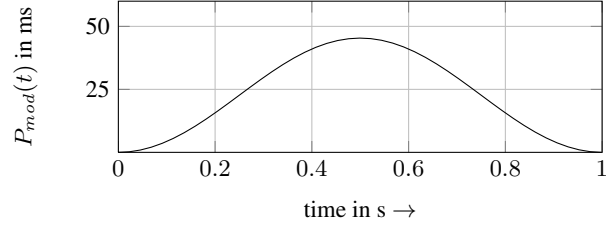
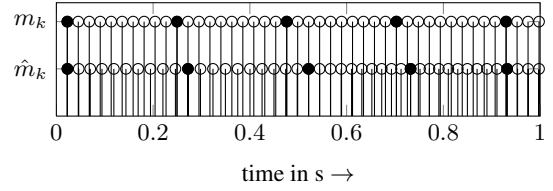


Figure 9: Example of varying delay simulation by PSOLA synthesis pitch mark modulation.

port the algorithms to C++ functions to use them as VST plugins without much effort.

For evaluation we used a test song with a male singer accompanied by an acoustic guitar. The tracks were available as separate lead vocal and guitar signals. An objective assessment of such system's performance is hard to realize. Therefore a subjective assessment of the harmonization results was conducted by a group of musicians having experience in that style of music. The synthesized backing vocals were found to be in good accordance with the guitar harmonies.

There are some artifacts perceivable as glitches which are mainly caused by wrong lead vocal pitch detection. These artifacts could be further reduced by smoothing of the pitch transitions. The hard assignment of the lead vocal pitches to the chord harmonies may also lead to glitches. An algorithm which allows a soft decision region could resolve this problem.

Some audio clips of the harmonization results can be found at <http://ant.hsu-hh.de/dafx2011/harmonization>.

6. CONCLUSIONS

We presented a system which harmonizes backing vocals based on the detected chords of an accompanying instrument. We proposed a modification of the Tolonen multipitch detector. The results show that the accuracy of multiple pitch detection and consequently of the chord classification for recorded guitar samples could be increased. The harmonization is operating completely autonomously, which means no key has to be manually provided. The developed algorithms operate in realtime which allows the use of the harmonization as live effect. The achieved harmonization results are quite promising but there is room for further improvement.

Future work will concentrate on the autonomous harmonization, since the presented voice leading approach is rather simple compared to how a real musician could harmonize. Also the chord detection will be extended to be able to detect seventh chords to improve harmonization capabilities.

7. REFERENCES

- [1] N. Schnell, G. Peeters, S. Lemouton, and X. Rodet, "Synthesizing a choir in real-time using Pitch Synchronous Overlap Add (PSOLA)," in *Proc. IEEE 1st Benelux Workshop on Model based Processing and Coding of Audio*, Leuven, 2002.
- [2] J. Bonada, M. Blaauw, A. Loscos, and K. Hideki, "Unisong: A choir singing synthesizer," in *Proc. 121th Audio Eng. Soc. Convention*, San Francisco, 2006.
- [3] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *J. Acoust. Soc. Am.*, vol. 111, no. 4, pp. 1917–1930, 2002.
- [4] A. von dem Knesebeck and U. Zölzer, "Comparison of pitch trackers for real-time guitar effects," in *Proc. 13th Int. Conf. on Digital Audio Effects (DAFx)*, Graz, 2010, pp. 266–269.
- [5] T. Tolonen and M. Karjalainen, "A computationally efficient multipitch analysis model," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 6, pp. 708–716, 2000.
- [6] R. Meddis and L. O'Mard, "A unitary model of pitch perception," *J. Acoust. Soc. Am.*, vol. 102, pp. 1811–1820, 1997.
- [7] K. Lee and M. Slaney, "Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 291–301, 2008.
- [8] H. Papadopoulos and G. Peeters, "Simultaneous estimation of chord progression and downbeats from an audio file," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, NV, 2008, pp. 121–124.
- [9] T. Cho and J. P. Bello, "Real-time implementation of HMM-based chord estimation in musical audio," in *Proc. Int. Computer Music Conference (ICMC 2009)*, Montreal, Canada, 2009.
- [10] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in *Proc. Int. Computer Music Conference (ICMC 1999)*, Beijing, China, 1999, pp. 464–467.
- [11] M. Cremer and C. Derboven, "A system for harmonic analysis of polyphonic music," in *Proc. 25th Int. Audio Eng. Soc. Conf.*, London, 2004.
- [12] C. Harte and M. Sandler, "Automatic chord identification using a quantised chromagram," in *Proc. 118th Audio Eng. Soc. Convention*, Barcelona, 2005.
- [13] A. M. Stark and M. D. Plumbley, "Real-time chord recognition for live performance," in *Proc. 2009 Int. Computer Music Conf. (ICMC 2009)*, Montreal, Canada, 2009.
- [14] C. Hamon, E. Mouline, and F. Charpentier, "A diphone synthesis system based on time-domain prosodic modifications of speech," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Glasgow, 1989, pp. 238–241.
- [15] K. Lent, "An efficient method for pitch shifting digitally sampled sounds," *Computer Music Journal*, vol. 13, pp. 65–71, 1989.
- [16] E. Moulines and F. Charpentier, "Pitch synchronous waveform processing techniques for text-to speech synthesis using diphones," *Speech Communication*, vol. 9, no. 5/6, pp. 453–467, 1990.
- [17] R. Bristow-Johnson, "A detailed analysis of a time-domain formant-corrected pitch-shifting algorithm," *J. Audio Eng. Soc.*, vol. 43, no. 5, pp. 340–352, 1995.
- [18] A. von dem Knesebeck, P. Ziraksaz, and U. Zölzer, "High quality time-domain pitch shifting using PSOLA and transient preservation," in *Proc. 129th Audio Eng. Soc. Convention*, San Francisco, 2010.

PHANTOM SOURCE WIDENING WITH DETERMINISTIC FREQUENCY DEPENDENT TIME DELAYS

Franz Zotter, Matthias Frank, Georgios Marentakis, Alois Sontacchi*

Institute of Electronic Music and Acoustics
University of Music and Performing Arts, Graz, Austria
{zotter, frank, marentakis, sontacchi}@iem.at

ABSTRACT

We present a novel method to adjust the perceived width of a phantom source by varying the deterministic inter channel time difference (*ICTD*) in a pair of signals over frequency. In contrast to given literature that focuses on random phase over frequency, our paper considers a deterministic approach that is open to a more systematic evaluation. Two allpass structures are described, finite impulse response (FIR) and infinite impulse response (IIR), for phase-based phantom source widening and evaluated in a formal listening test. Varying *ICTD* over frequency essentially alters the inter-aural cross correlation coefficient at the ears of a listener and in this way provides a robust way to control the auditory source width. The subjective evaluation results fully support our observations for both noise and speech signals.

1. INTRODUCTION

Two loudspeakers emitting the same sound simultaneously will create the illusion of a phantom source, a sound localized in between. A long lasting problem in the field is the manipulation of the perceived phantom source width. Up until now, the main ways to manipulate it were the so-called pseudo stereo or decorrelation approaches.

Purely phase-based decorrelation techniques were comprehensively studied in the work of Kendall [1], which gives an overview of already convincing perceivable effects, using random phase values for different frequencies. Improvements and variations were described in [2, 3]. The control of a random phase is nevertheless a challenging task and evaluation results verify the difficulty inherent in reducing the variability in the perceptual outcome of such a process.

Pseudo stereo can be achieved by various kinds of implementations, which essentially introduce fluctuations in the frequency dependent level and phase differences of a pair of playback signals. Schröder [4] and Orban [5, 6] describe the Lauridsen network which produces a pair of spectrally complementing comb filters. Gerzon [7] discusses that previous work on Lauridsen networks suffers from phasing, a serious side-effect. He proposed various alternative strategies to reduce this effect: a) frequency

varying amplitude panning, b) a modified Lauridsen network with a delayed channel, and c) unitary feedback delay networks.

Of the techniques proposed by Gerzon, frequency varying amplitude panning is very close to observations made independently by Blauert and Lindemann. In their work, they mention [8] that panning of individual frequency bands in specific directions, or the rapid interchange of the panning directions of the frequencies within a signal, reduces the inter-aural cross correlation coefficient (IACC) and increases the phantom source width, i.e., auditory source width (ASW). They also verify the inverse relation between IACC and ASW for headphone listening by combining uncorrelated noise sources. The IACC is the maximum of the normalized inter-aural cross correlation function within a maximum time shift of $\pm 1\text{ms}$ [9]. Despite the evaluation using headphones, their work is fundamental when creating a pseudo-stereo or decorrelated signal pair out of a single monophonic sound.

The main focus of this work is to revise phase-based approaches that neither suffer from phasing nor require special skill in designing random variables by taking into account the observations by Blauert and Lindemann [8] and Gerzon [7]. To this end, we establish the relationship between the phase of an allpass filter and its frequency dependent group delay. Using the group delay differences in a pair of allpass filters enables the deterministic control of *ICTD*. Suitable implementations are provided for both FIR and IIR filter designs. Eventually, the target phase is determined by an accurately reproducible frequency dependent inter channel time difference (*ICTD*). This approach is equivalent to a frequency dependent time delay panning in stereophony. The perceived phantom source widening using the presented algorithms is evaluated and verified by a listening experiment and related to the IACC, using a setup illustrated in Fig. 1.

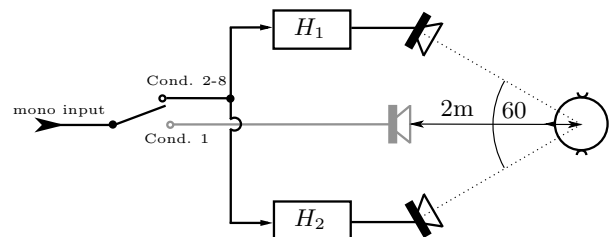


Figure 1: The setup used in the evaluation experiment.

* Contributions: Franz Zotter developed two allpass structures yielding frequency dependent *ICTD* for decorrelation. Georgios Marentakis proposed to investigate ASW control through frequency dependent panning. Matthias Frank and Georgios Marentakis designed the evaluation study executed by Matthias Frank. Georgios Marentakis and Matthias Frank executed the statistical analysis. Alois Sontacchi contributed his expertise on the field of signal decorrelation, perceptual evaluation and in the presentation of research.

2. ALLPASS STRUCTURES FOR DETERMINISTIC ICTD

Decorrelation [1] designs are usually done by means of a random allpass phase $\phi[k]$ in the discrete Fourier transform (DFT) domain. Introducing random phase in a signal cannot be done without constraints imposed to maintain acceptable sound quality.

Assuming an arbitrarily designed phase pair ϕ_1, ϕ_2 , the out of phase signals are easily avoided by forbidding these values in the design, i.e. $\phi_2 - \phi_1 \neq (2l + 1)\pi, l \in \mathbb{Z}$.

Another issue is to ensure that the magnitude response stays unity between the frequency bins k , cf. [1], or similarly, that the impulse response does not suffer from cyclic time-domain aliasing. The difference of the random phase in successive bins is therefore subjected to a limitation $|\phi[k] - \phi[k - 1]| < \Delta\phi_{\max}$. After this limitation, the achievable effect is mainly controlled by this limit and the DFT-length N . In [3] it is even proposed to make this limit depending on the Bark frequency to improve some perceptual qualities.

This phase limitation is quite interesting for the deterministic design as there is an underlying meaning that is useful. The (cyclic) group delay in one frequency bin can be estimated using the first backward difference of the phase $\tau[k] = \frac{N}{2\pi f_s} \Delta\phi[k]$, with the signal sampling frequency f_s . Hence, the limitation of the phase change over successive bins is observed to limit the group delay, implicitly. But it also means that the group delay $\tau[k]$ could be designed as a random variable to construct the phase

$$\phi[k] = \frac{2\pi f_s}{N} \sum_{k'=0}^k \tau[k']. \quad (1)$$

Nevertheless, the limitation of frequency dependent random group delays might not be enough as a design parameter for interesting effects influencing the phantom source image. For instance, the group delay should change often enough over frequency to avoid angular shifts of the phantom source. On the other hand, too frequent changes might cause impulse responses of inconvenient length. Therefore, a cosine function is proposed to design the group delay.

2.1. Deterministic FIR allpass design

The presented study favors a deterministic FIR allpass design that is reproducible and has a controllable variation of the *ICTD* over frequency.

As the simplest choice of its deterministic behavior, a positive and negative cosine contour with adjustable frequency period Δf and peak value $\hat{\tau}$ can be chosen as the group delay of the transfer functions H_1, H_2

$$\tau_{1,2}(\omega) = \mp \hat{\tau} \cos(\omega/\Delta f), \quad (2)$$

with the angular frequency variable $\omega = 2\pi f$. This yields, with $\tau_2 - \tau_1$, an adjustable inter channel time delay

$$ICTD(\omega) = 2\hat{\tau} \cos(\omega/\Delta f). \quad (3)$$

By the negative integral of the group delay over ω , the sinusoidal phase $\phi(\omega) = \pm \hat{\tau} \Delta f \sin(\omega/\Delta f)$ of the allpass decorrelation filters are obtained, with $j = \sqrt{-1}$,

$$H_{1,2}(\omega) = e^{\pm j \hat{\tau} \Delta f \sin(\omega/\Delta f)}. \quad (4)$$

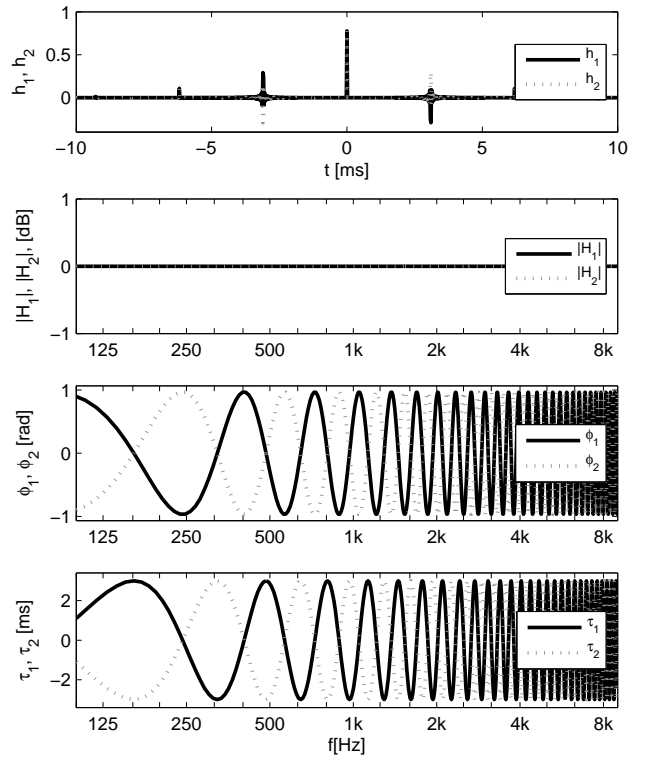


Figure 2: Impulse, magnitude, phase, and group delay responses of the deterministic FIR allpass pair.

It is not surprising that the structure of the corresponding impulse response is similar to an FM-spectrum with its Bessel functions $J_m(\mu)$ of various orders m and the modulation depth μ . For the two allpass functions $h_1(t)$ and $h_2(t)$ with opposing time delays, the following impulse responses are obtained

$$h_{1,2}(t) = \sum_{m=-\infty}^{\infty} J_{\pm m}(\hat{\tau} \Delta f) \delta\left(t - \frac{m}{\Delta f}\right). \quad (5)$$

The corresponding inter channel time delay is adjustable with regard to the frequency period $\Delta f/2$ in which the sign of the *ICTD* alternates its magnitude between $-\hat{\tau} \leq ICTD \leq \hat{\tau}$. The FIR implementation is made causal by introducing a suitable time delay. Its length is limited as Bessel functions vanish with large $|m|$ and small $\hat{\tau} \Delta f$. Fig. 2 shows the impulse, magnitude, phase, and group delay responses of the system. In order to avoid opposite inter channel phase, the phase argument of one channel ϕ needs to be restricted to $\pm\pi/2$, i.e.

$$\hat{\tau} \Delta f < \frac{\pi}{2}. \quad (6)$$

2.2. IIR allpass design

As an alternative IIR implementation of the phantom source widening effect, an implementation with third-octave structure can be defined. Let us assume two cascade chains $l = 1, 2$ containing

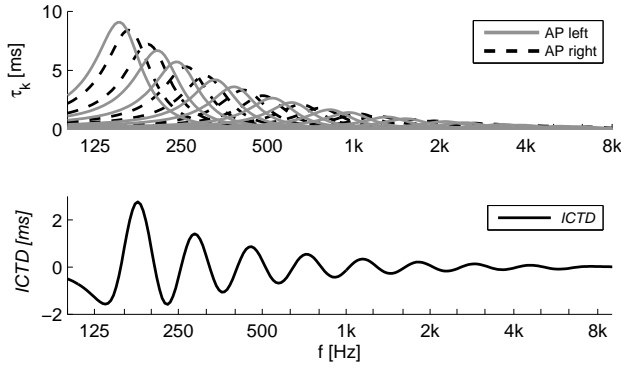


Figure 3: Group delay responses of the 2nd order IIR allpass chain elements for the channel 1 and 2 and the achieved *ICTD*.

third-octave 2nd order allpass filters

$$H_l(s) = \prod_{k=-8}^8 H_{k,l}(s), \quad l = 1, 2. \quad (7)$$

The allpass k in the cascade l is, for simplicity, described in the Laplace/Fourier domain as a complex continuous-time filter

$$H_{k,l}(s) = \frac{s - j\omega_{k,l} - \sigma_{k,l}}{s - j\omega_{k,l} + \sigma_{k,l}} \Big|_{s=j\omega} = e^{j2 \arctan \frac{\omega_{k,l} - \omega}{\sigma_{k,l}}}. \quad (8)$$

We may use a reference angular frequency $\omega_0 = 2\pi$ 1kHz in order to define the center frequencies of the filters $\omega_k = 2^{k/3} \omega_0$ and their bandwidths $\sigma_k = \frac{\partial}{\partial k} \omega_k = \omega_k / Q$ with $Q = 1 / \ln 2^{1/3}$. Two identical cascades do not create any *ICTD* yet, but only require slight modifications to do so.

A frequency varying *ICTD* is obtained when the center frequencies of both chains are shifted alternatingly towards lower and higher frequencies by some factors ϵ^{-1} or ϵ , respectively. Defined accordingly, the allpass parameters of the chain l are

$$\omega_{k,l} = \epsilon^{(-1)^{k+l}} 2^{k/3} \omega_0, \quad (9)$$

$$\sigma_{k,l} = \omega_{k,l} / Q. \quad (10)$$

For good results, ϵ should be bounded $1 \leq \epsilon < 2^{1/6}$ to avoid frequencies with opposite inter channel phase.

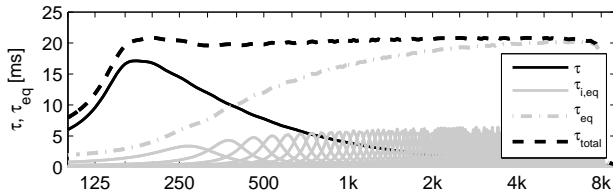


Figure 4: Group delay response of the allpass chain and its equalization with 150 2nd order IIR allpasses, and the compensated group delay curve.

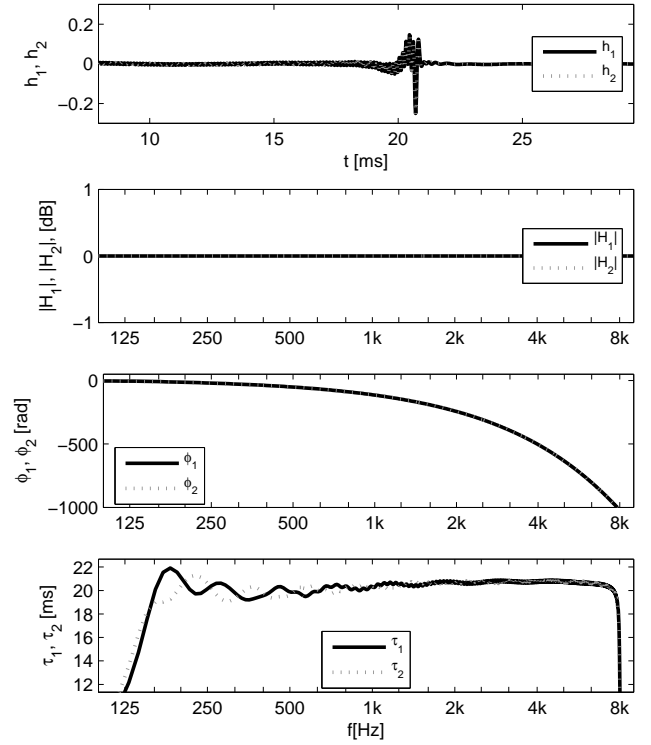


Figure 5: Impulse, magnitude, phase, and group delay responses of the deterministic IIR allpass chain pair out of third-octave 2nd order allpasses and their 150 element group delay compensation.

A closer insight into the proposed structure is obtained by regarding the group delay. The negative derivative of the phase is easy to calculate for one element k , and, in sum, yields the group delay introduced by the cascade l

$$\tau_l(\omega) = \sum_{k=-8}^8 \frac{2\sigma_{k,l}^2}{\sigma_{k,l}^2 + (\omega_{k,l} - \omega)^2}. \quad (11)$$

The *ICTD* is the difference between the group delays of both cascades

$$ICTD(\omega) = \tau_2(\omega) - \tau_1(\omega). \quad (12)$$

Essentially, this achieves the desired effect, see bottom diagram in Fig. 3, but the group delay distortion of a third-octave allpass cascade is clearly audible and slightly annoying for speech and transients. An equalizer for the group delay has been designed to avoid this in the proposed implementation, see Fig. 4. This is a 150 element allpass chain, designed to compensate for the group delay curve

$$\tau_{eq}(\omega) = \tau_c - \frac{\tau_1(\omega) + \tau_2(\omega)}{2} \quad (13)$$

up to a constant offset τ_c . For more details on the design of allpass chains of 2nd order elements with specific overall dispersion behavior see [10]. Fig. 3 shows the group delay responses of all the IIR cascade elements, its sum, the dispersion compensation allpasses and the overall obtained *ICTD*.

	Description	IACC	IACC _{E3}
C1	real source	0.886	0.875
C2	phantom source	0.824	0.815
C3	IIR allpass	0.741	0.725
C4	FIR allpass, $\hat{\tau} = 3\text{ms}$, $\Delta f = 200\text{Hz}$	0.715	0.690
C5	FIR allpass, $\hat{\tau} = 1\text{ms}$, $\Delta f = 1200\text{Hz}$	0.528	0.559
C6	FIR allpass, $\hat{\tau} = 3\text{ms}$, $\Delta f = 400\text{Hz}$	0.598	0.554
C7	FIR allpass, $\hat{\tau} = 6\text{ms}$, $\Delta f = 200\text{Hz}$	0.571	0.521
C8	FIR allpass, $\hat{\tau} = 3\text{ms}$, $\Delta f = 600\text{Hz}$	0.394	0.468

Table 1: Description of the tested conditions.

In Fig. 5 the resulting impulse, magnitude, phase, and group delay responses are given. In this implementation, the parts of the impulse responses lying below 200Hz and above 7kHz were left untreated and come without delay for simplicity of illustration.

3. EVALUATION

We created eight conditions (described in Table 1) that correspond to the discussed techniques and evaluated them in a formal listening test. The first two conditions were control conditions, included to verify that the manipulations yield a larger source width relative to an untreated phantom source (C2) and a mono source (C1). The IIR condition C3 does not have many degrees of freedom thus it was evaluated in a single condition. C4-C8 are more versatile and were tested with various parameter settings for Δf and $\hat{\tau}$. The IACC for our eight conditions was computed from dummy head recordings using a B&K 4128C in the real listening test setup. For completeness, a recent measure proposed in [11], the IACC_{E3} was also computed. IACC_{E3} is the mean of the IACC computed in 3 octave bands (500Hz, 1KHz and 2KHz) for the first 80ms.

3.1. Method

Participants were seated according to Figure 1 and were facing forward, i.e. the participants were facing the centre of the sound event. They were presented with all possible pairwise comparisons of the eight experimental conditions and indicated which sound in a pair (A, B, or none) is wider by pressing the corresponding buttons on a keyboard. They could listen and switch between the sounds in a pair at will.

The test was divided into two parts with different stimuli: 5s of pink noise in the first and speech in the second part. For the second part, a mono sample of 22s male English speech was used from the EBU SQAM CD [12]. Stimuli were presented at 65dB(A) for noise and 65 L_{eq}(A) for speech. Before each part, a short training phase was conducted to familiarize the participants with the comparison task and the stimuli. Each of the (8 choose 2 =) 28 paired comparisons was rated twice by one participant, yielding a number of 56 comparisons for each part; a total number of 112 comparisons per participant for both parts. The order of presentation and the location of each stimulus within each pair (A, B) was randomized.

3.2. Setup

Three Genelec 8020 loudspeakers were placed at -30°, 0°, and +30°, 2m from the participants' head, see Figure 1. The size of

	C1	C2	C3	C4	C5	C6	C7	C8
C1	-/-	91/84	91/100	95/84	100/100	100/100	95/100	100/95
C2	9/16	-/-	82/91	86/68	95/100	100/95	100/100	100/100
C3	9/0	18/9	-/-	75/48	91/77	91/93	100/100	100/100
C4	5/16	14/32	25/52	-/-	68/66	91/93	86/98	100/100
C5	0/0	5/0	9/23	32/34	-/-	77/86	86/89	100/95
C6	0/0	0/5	9/7	9/7	23/14	-/-	48/50	91/75
C7	5/0	0/0	0/0	14/2	14/11	52/50	-/-	82/70
C8	0/5	0/0	0/0	0/0	0/5	9/25	18/30	-/-

Table 2: Relative frequency dominance matrix containing the average percent of times participants responded that a column is wider than a row for both noise/speech stimuli; numbers are rounded integers of the relative votes in %.

the playback room was approx. 11m × 11m × 5m. The average reverberation time RT₆₀ was 470 ms. Although with reference to ITU-R BS.1116-1 [13] the room is large, it still is within the recommended reverberation time limits. In addition, participants were seated within the effective critical distance of the setup, which was calculated to be 2.76m. 11 participants (4 female, 7 male) participated in the listening test (age range: 23-32 years, median: 27 years). All participants were part of a trained listening panel [14, 15] and familiar with the evaluation of source width, as they had participated in another source width experiment [16].

We tested the following hypotheses: 1. The decreased IACC or IACC_{E3} created by the variation of the *ICTD* over frequency affects the perceived ASW. 2. The effect of the different conditions is consistent for different input signal types as long as the *ICTD* variability is distributed over the signal's bandwidth. 3. The perception of the ASW increases with an increase in the magnitude of the *ICTD*. 4. The perception of the ASW is affected by Δf .

3.3. Results

Participants responded consistently throughout the experimental trials. Only 11% of the repetitions for noise and 14% for speech contained a different response. Most of this variation was observed when they compared pairs C3/C4, and C6/C7. Table 2 shows the relative frequency dominance matrix averaged across all participants and repetitions. A first indication that the different conditions yielded different ASW impressions can be obtained by examining the percent of the responses in which participants judged condition C_{i+1} to be wider than C_i , $i = 1 \dots 7$. For noise, all comparisons yield percent discrimination significantly larger than chance ($p < 0.05$) apart from comparison C6/C7 and C4/C5. Results for speech stimuli were the same with the additional exception of condition comparison C3/C4.

A more complete picture can be obtained by creating psychophysical scales that take into account the participants' responses in all possible pairwise comparisons between the eight conditions. ASW scales were constructed based on the dataset in Table 2 according to Thurstone Case V [17] and the BTL model [18] and are presented in Table 3. Conditions were ranked in the same way by both models however the differences between the scale values of each condition varied. We attribute this discrepancy to the fact that our experimental conditions were easily distinguishable yielding ceiling effects that are handled differently in the two models. Although we concentrate our analysis on the Thurstone scales, our conclusions are supported by both models.

	Thurstone		BTL	
	Noise	Speech	Noise	Speech
C1	0	0	0	0
C2	0.19	0.14	0.0018	0.0043
C3	0.33	0.38	0.0063	0.0256
C4	0.48	0.30	0.0191	0.0183
C5	0.61	0.50	0.0398	0.0692
C6	0.78	0.87	0.2648	0.3972
C7	0.82	0.89	0.1473	0.8785
C8	1	1	1	1

Table 3: Scale values normalized within [0,1] based on Table 2, using the Thurstone and BTL model.

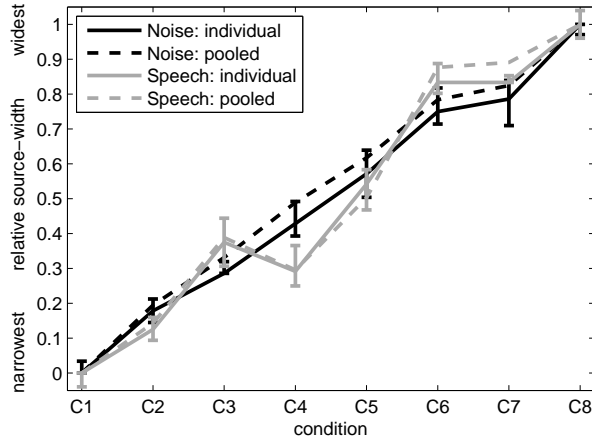


Figure 6: Thurstone scales for noise/speech: median and 95% CI using participants' individual scales compared to scales using answers pooled from all participants, Tab. 2. Scales were normalized within [0,1].

Thurstone scales for each subject were calculated based on each individual's frequency dominance matrix. It is worth noting that the median of the scales from the individuals' responses and the scales of the pooled responses constructed in the previous paragraph were almost identical (see Figure 6). A 2-way (Stimulus x Condition) ANOVA was then performed on the scale values of each individual as a function of the condition and stimulus type used in the experiment. Analysis of variance yielded no effect of stimulus type and only a rather weak interaction between stimulus and condition, $F(7,70) = 2.08$, $p = 0.057$. There was a significant main effect of condition, $F(7,70) = 263.616$, $p < 0.001$. Post-hoc comparisons using Bonferroni confidence interval adjustment showed that all conditions yielded significantly different source width perceptions ($p < 0.001$) with the exception of C3/C4 and C6/C7. In the latter case no difference was observed for both noise and speech. In the former, a significant difference was observed for noise but none for speech, attributing partially to the marginally significant interaction. The significant difference between conditions C4 and C7 indicates that ASW perception depends on the magnitude of $ICTDs$. One could try to predict the effect of Δf from the fact that C4 yielded significantly lower ASW impressions than both C6 and C8, implying an inverse relationship of ASW to Δf . However, this is not confirmed when comparing C6 and C7.

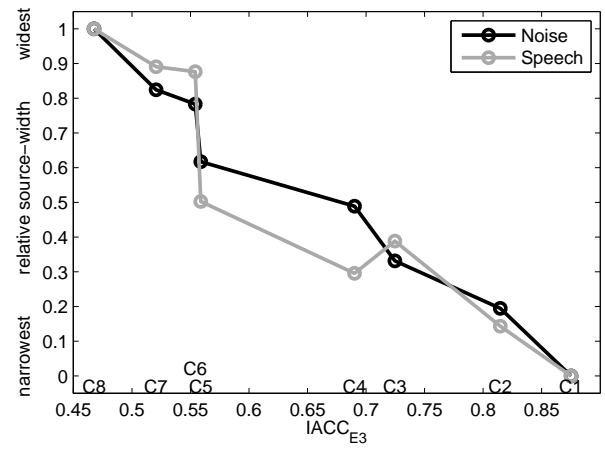


Figure 7: Thurstone scales for answers pooled from all participants as a function of the $IACC_{E3}$. Scales were normalized within [0,1], $IACC_{E3}$ of the conditions are indicated.

3.4. Discussion

Our hypotheses were in general confirmed by the experimental results. Hypothesis 1 stating that the decreased $IACC$ or $IACC_{E3}$ as a consequence of adjusting $ICTDs$ would yield different ASW impressions was confirmed by the significant main effect of condition observed in our experiment. The Thurstone scales from the pooled participants' responses correlated with $IACC$ at $r = 0.95$ for noise and $r = 0.92$ for speech. Similarly, $IACC_{E3}$ correlated at $r = 0.98$ and $r = 0.97$ with the noise and speech stimuli respectively, yielding a nearly linear relation as is clearly shown in Figure 7.

Hypothesis 2 stating that the manipulation of the $ICTD$ would apply to diverse stimuli types was confirmed for the noise and speech used in our experiment. The ASW scales already presented in Table 3 for noise and speech were highly correlated to each other ($r = 0.96$). However, further studies are required to fully confirm this hypothesis for other signal types.

Hypothesis 3 stating that the ASW impression would depend on the magnitude of the $ICTD$ was confirmed by the significant difference between conditions C4 and C7.

Hypothesis 4 was also confirmed, however, a proper identification of the effect of Δf will have to be discussed in future studies.

4. CONCLUSION

We have presented methods for phantom source widening using deterministic frequency dependent time delays. The methods have been implemented as deterministic FIR and IIR allpass structures. The presented algorithms are useful new audio effects for spatial sound imaging with well controllable behavior. However, clearly, the comprehensive evidence of all perceived aspects and comparison to earlier ideas about pseudo-stereo are outside the scope of this paper and remain subject to future studies. It is worth noting that in confirmation of the authors' subjective impression from listening to the presented algorithms, participants did not report annoyance or timbral deficiencies in the listening experiments. This however remains to be established in a more formal way in future

studies.

Importantly, we show that frequency dependent *ICTD* panning can be successfully used to widen a phantom source. Although our evaluation results are on *ICTD* panning, informal experimentation shows that the same technique can also be used with frequency dependent *ICLD* panning. Frequency dependent panning works essentially because it adjusts the inter-aural cross correlation coefficient of the received signal and therefore provides a robust way to control apparent sound source width. Several issues remain unresolved however. In particular, it would be interesting to identify when exactly image splitting starts to appear and the role of different panning envelopes and individual frequencies. Such undertakings will likely increase our knowledge about the mechanisms that leads to robust ASW representation in the brain and assist us in the creation of better ways to control ASW.

5. ACKNOWLEDGMENTS

This project is partly funded by the Austrian Research Promotion Agency (FFG), the Styrian Government and the Styrian Business Promotion Agency (SFG) under the COMET programme. Thanks to all participants for their participation and to the project partners for assistance.

6. REFERENCES

- [1] G. S. Kendall, "The decorrelation of audio signals and its impact on spatial imagery," *Computer Music J.*, vol. 19, no. 4, 1995.
- [2] G. Potard and I. Burnett, "Decorrelation techniques for the rendering of apparent sound source width in 3d audio displays," in *Proc. DAFx-07*, Napoli, 2007.
- [3] M. Bouéri and C. Kyriakakis, "Audio signal decorrelation based on a critical band approach," in *Preprint 117th Conv. Audio Eng. Soc.*, San Francisco, 2004.
- [4] M. R. Schröder, "An artificial stereophonic effect obtained from a single audio signal," *J. Audio Eng. Soc.*, vol. 6, no. 2, 1958.
- [5] R. Orban, "A rational technique for synthesizing pseudo-stereo from monophonic sources," *J. Audio Eng. Soc.*, vol. 18, no. 2, 1970.
- [6] R. Orban, "Letters to the editor: Further thoughts on 'a rational technique for synthesizing pseudo-stereo from monophonic sources'," *J. Audio Eng. Soc.*, vol. 18, no. 4, 1970.
- [7] M. A. Gerzon, "Signal processing for simulating realistic stereo images," in *Preprint 3423, 93rd Conv. Audio Eng. Soc.*, San Francisco, 1992.
- [8] J. Blauert and W. Lindemann, "Spatial mapping of intracranial auditory events for various degrees of interaural coherence," *The Journal of the Acoustical Society of America*, vol. 79, pp. 806–813, 1986.
- [9] ITU, "ISO 3382:1997 acoustics - measurement of the reverberation time of rooms with reference to other acoustical parameters," Tech. Rep., 1997.
- [10] J. S. Abel and J. O. Smith, "Robust design of very high-order allpass dispersion filters," in *Proc. DAFx-06*, Montreal, 2006.
- [11] T. Hidaka, L. L. Beranek, and T. Okano, "Interaural cross-correlation, lateral fraction, and low- and high-frequency sound levels as measures of acoustical quality in concert halls," *J. Acoustical Soc. Am.*, vol. 98, 2, pp. 988–1007, 1995.
- [12] EBU, "Tech 3253 - sound quality assessment material (sqam)," <http://tech.ebu.ch/publications/sqamcd>.
- [13] ITU, "ITU-R BS.1116-1 methods for the subjective assessment of small impairments in audio systems including multichannel sound systems," Tech. Rep., 1994-1997.
- [14] A. Sontacchi, H. Pomberger, and R. Höldrich, "Recruiting and evaluation process of an expert listening panel," in *Fortschritte der Akustik, NAG/DAGA*, Rotterdam, 2009.
- [15] M. Frank, A. Sontacchi, and R. Höldrich, "Training and guidance tool for listening panels," in *Fortschritte der Akustik, DAGA*, Berlin, 2010.
- [16] M. Frank, G. Marentakis, and A. Sontacchi, "A simple technical measure for the perceived source width," in *Fortschritte der Akustik, DAGA*, Düsseldorf, 2011.
- [17] L. L. Thurstone, "A law of comparative judgment," *Psychological Review*, vol. 101, no. 2, pp. 266–270, 1994.
- [18] F. Wickelmaier and C. Schmid, "A matlab function to estimate choice model data from paired-comparison data," *Behavior Research Methods, Instruments and Computers*, vol. 36, no. 1, pp. 29–40, 2004.
- [19] C. Faller, "Parametric coding of spatial audio," in *Proc. DAFx-04*, Napoli, 2004.

IMPLEMENTING REAL-TIME PARTITIONED CONVOLUTION ALGORITHMS ON CONVENTIONAL OPERATING SYSTEMS

Eric Battenberg, Rimas Avižienis

Parallel Computing Laboratory
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA
{erich, rimas}@eecs.berkeley.edu

ABSTRACT

We describe techniques for implementing real-time partitioned convolution algorithms on conventional operating systems using two different scheduling paradigms: time-distributed (cooperative) and multi-threaded (preemptive). We discuss the optimizations applied to both implementations and present measurements of their performance for a range of impulse response lengths on a recent high-end desktop machine. We find that while the time-distributed implementation is better suited for use as a plugin within a host audio application, the preemptive version was easier to implement and significantly outperforms the time-distributed version despite the overhead of frequent context switches.

1. INTRODUCTION

Partitioned convolution is a technique for efficiently performing time-domain convolution with low inherent latency[1]. It is particularly useful for computing the convolution of audio signals with long (>1 second) impulse responses in real time, as direct convolution becomes too computationally expensive and block-FFT convolution incurs unacceptable latency.

Much of the existing work on partitioned convolution has focused on optimizing the computational pieces of the algorithm, i.e. efficiently implementing FFTs and spectral multiplications[1][2][3] and finding computationally optimal partitionings[4]. In this paper, we focus on how to most effectively schedule the necessary computations on a personal computer using a conventional operating system.

We investigate the performance of two scheduling approaches for non-uniform partitioned convolution: a multi-threaded, preemptive approach and a cooperative, time-distributed approach. Issues we explore include performance, compatibility with existing audio hosts, and programming effort.

In Section 2, we cover the basics of non-uniform partitioned convolution. Sections 3 and 4 cover the implementation of the preemptive and cooperative approaches, respectively, and Sections 5 and 6 present and discuss the performance results and their implications.

2. ALGORITHM OVERVIEW

Convolution is a mathematical operation commonly used to perform finite impulse response (FIR) filtering on a signal:

$$y[n] = \sum_{k=0}^{L-1} x[k]h[n-k] \quad (1)$$

Where x and y are the input and output signals, respectively, and h is the length- L impulse response of the FIR filter.

The above direct method of convolving two signals has no inherent latency but carries with it a large computational cost per output sample ($O(L)$ multiply-adds per output sample). Because of this, real-time convolution with larger impulse responses is usually carried out using block FFT-based methods, like overlap-add and overlap-save. These methods take the FFTs of the impulse response and a buffered portion of the input signal, multiply them together in the frequency domain, and take the inverse FFT of their complex product to compute a portion of the output signal [5]. Computing convolution in this way requires significantly less computation ($O(\log L)$) at the expense of increased latency due to buffering.

2.1. Uniform Partitioned Convolution

In order to obtain a compromise between computational efficiency and latency, we can partition the impulse response into a series of smaller sub-filters which can be run in parallel with appropriate delays inserted. Each sub-filter's output is computed using a block-FFT method, and the outputs of all sub-filters are summed to produce a block of the output signal, as shown in Figure 1.

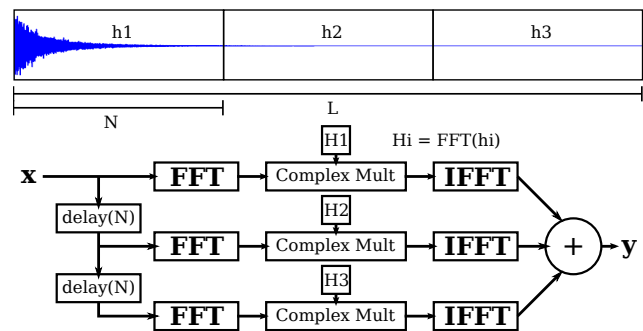


Figure 1: Top – Partitioning of an impulse response into 3 parts. Bottom – Steps involved in computing the above 3-part uniform partitioning.

If the original length- L filter is partitioned into sub-filters of size N , we perform $O(\log N)$ operations per sub-filter per output sample but have reduced the latency from L to N . This scheme works well but can become infeasible if low latency is required for filters longer than a second or two. For example, if 1.5ms processing latency is desired (64 samples at 44.1kHz), a 92ms (4096

sample) filter would need to be cut into 64 sub-filters, while a 6sec (262144 sample) filter would require 4096 sub-filters.

Within this uniform partitioning scheme, we can save previously computed forward FFTs for reuse in subsequent sub-filters. Additionally, linearity of the FFT allows us to sum the complex frequency domain output of each sub-filter before taking the inverse FFT, which reduces the number of inverse FFTs required to one. In Figure 1, these optimizations would be made by replacing the FFT/IFFTs with a single FFT before the delays and a single IFFT after the sum as shown in Figure 2. Because we are now delaying frequency-domain data, Garcia refers to this method of computing a uniform partitioning as a *Frequency-domain Delay Line* (FDL) [4].

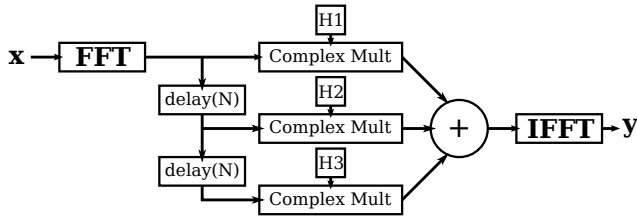


Figure 2: *Frequency-domain Delay Line (FDL)*

Even with this FDL optimization, the computational cost of uniform partitioned convolution (influenced primarily by the number of complex multiplications and additions of frequency domain coefficients) scales linearly with the impulse response length and its use becomes impractical for very long impulse responses.

2.2. Non-Uniform Partitioned Convolution

Non-uniform partitioned convolution attempts to improve upon the computational efficiency of the uniform partitioned convolution method by dividing the impulse response into partitions of various sizes. The approach is to use shorter partitions near the beginning of the impulse response to achieve low latency and longer partitions towards the end to take advantage of increased computational efficiency. In [1], Gardner describes how to use such a mix of partitions sizes to improve efficiency without sacrificing latency.

Gardner suggests a partitioning scheme that increases the partition size as quickly as possible, as shown at the top of Figure 3. However, Garcia [4] points out that since the FDL optimization can be applied to each block size used in a non-uniform partitioning, it is usually more efficient to use more partitions of a given size (bottom Figure 3) before moving to a larger partition size [1]. Therefore, we can view a non-uniform partitioning as a parallel composition of FDLs of increasing block size.

In longer FDLs, execution time is dominated by complex multiplication and summation. These operations can be optimized by viewing the complex multiplications as convolutions performed in the frequency domain and applying techniques described by Hurchalla [3] for efficiently performing running convolutions with short to medium length sequences. It is also possible to reduce FFT-related computations by computing the larger FFTs from the intermediate values of smaller FFTs [1]. Additional computational improvements include enhancing FFT and complex arithmetic efficiency via system-specific optimizations such as using SIMD instructions and cache-aware tuning [6].

FDL-based non-uniform partitioned convolution is a very computationally efficient approach to low-latency real-time convolu-

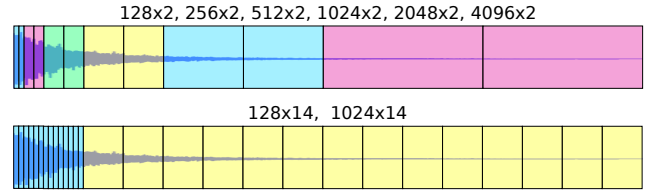


Figure 3: *Two non-uniform partitionings of an impulse response of length 16128 with $N = 128$. Top – Gardner partitioning with 6 FDLs. Bottom – Optimal Garcia partitioning with 2 FDLs.*

tion; however, a real-world implementation of this approach still leaves many decisions to be made, the most important being: how should we schedule all of this computation on a conventional operating system? We cover two approaches to scheduling in the following two sections.

3. PREEMPTIVE IMPLEMENTATION

A non-uniform partitioning consists of multiple FDLs which execute concurrently but perform their processing during different time periods. The shortest period, which should be equal to the audio callback interval, is associated with the primary FDL. Subsequent FDLs use larger block sizes and therefore have longer periods. In order to avoid having to process longer FDLs within a single callback interval, Gardner suggests that longer FDLs be allowed to run for a time interval equal to their period, rather than within a single callback period. This helps to preserve uniform processor loading. Figure 4 illustrates processing boundaries in time (arrivals and deadlines) for a partitioning with 3 FDLs.

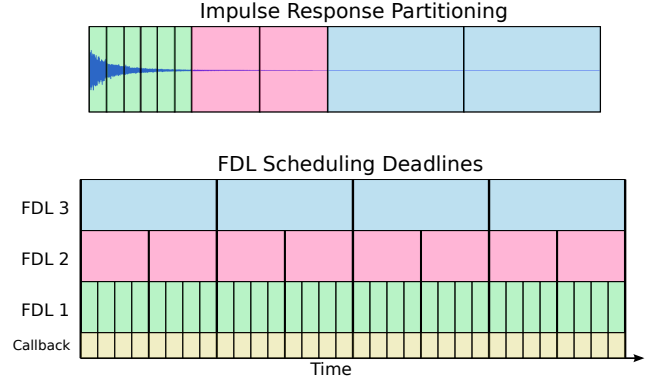


Figure 4: *Top – Example non-uniform partitioning with 3 FDLs. Bottom – Scheduling boundaries of FDL tasks. Arrivals/deadlines are denoted by vertical lines.*

3.1. Computation

In order to maintain a dropout-free audio stream, we must ensure that all of an FDL's computations are complete by its processing deadline. In our case, the bulk of the processing time is spent computing FFTs and performing complex arithmetic. A popular choice for a portable FFT library is FFTW[7], which performs well on a wide variety of platforms by performing an “auto-tuning” step where it measures the performance of many FFT sub-routines and picks the fastest combination for the target system.

For longer FDLs (those containing more partitions), the complex multiply-add (Cmadd) routine becomes the computational bottleneck. We implemented several versions of the Cmadd routine, with the performance of each shown in Figure 5. The slowest routine uses the built-in complex type defined in GCC’s `complex.h` in a simple for-loop. Our “naive” version computes the real and imaginary components in separate lines of code. The “SIMD” version makes full use of Intel’s SSE/SSE3 SIMD instructions. The “naive” and “SIMD” versions were further optimized using 4x manual loop unrolling (“+unroll”), which reduces indexing arithmetic and helps the compiler better take advantage of instruction level parallelism. The “complex.h” version saw no improvement from loop unrolling. Performance numbers in this section were produced using Apple’s version of GCC 4.2 and our Mac OS X 10.6 test platform, which consists of a MacBook Pro running a 2.66GHz Intel Core 2 (Penryn) processor. On this platform, the “SIMD+unroll” routine performed 4x–15x faster than the `complex.h` routine and 2x–8x faster than the naive routine

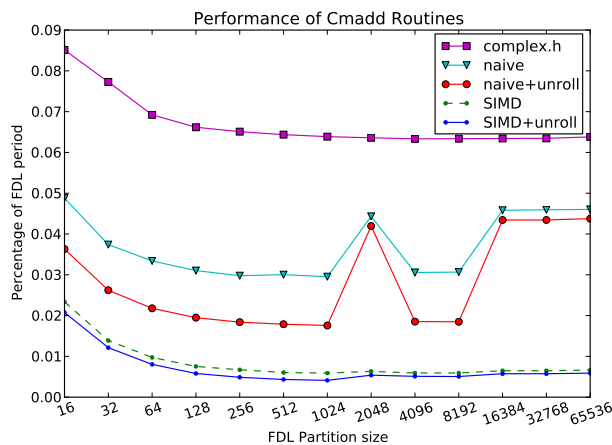


Figure 5: Cmadd routine running time as a percentage of FDL period

To put the absolute execution times of these routines in perspective, for an audio I/O buffer size of 32 samples at 44.1kHz, the callback interval is 725 μ s. A first level FDL will do one FFT and one inverse FFT each of size 64, and it will run the Cmadd routine once for every partition in the FDL. On our Mac test platform, FFTW’s out-of-box forward and reverse FFT routines take 1.1 μ s and 0.32 μ s respectively, while the routines chosen using “FFTW_PATIENT” auto-tuning take 0.27 μ s and 0.25 μ s. The naive Cmadd routine takes 0.18 μ s per partition, while the optimized routine takes 0.075 μ s. At this block size, the optimized Cmadd routine allows this first-level FDL to handle more than two times as many partitions with the same processor load.

Finally, buffering operations in our Linux implementation greatly benefited from the use of *asmlib*[8], which includes optimized memory movement routines (`memset`, `memcpy`, etc) which can perform up to 10x faster than the default versions used by `glibc` for properly aligned memory regions.

3.2. Scheduling

As shown in Figure 4, the FDLs run concurrent tasks with different execution periods and deadlines. Because of this, we run each

FDL in its own thread, which allows an FDL with an earlier deadline to preempt one with a later deadline. However, as we will discuss later, the inclusion of preemptive multi-threading within this implementation can cause problems when sharing computing resources with other audio processing tasks.

In our implementation, we use the POSIX threads (*pthread*s) API[9], to create and manage the execution and scheduling of worker threads. For each FDL, we create a worker thread that is responsible for executing the FFTs and complex arithmetic required by the FDL. Synchronization of the worker threads and buffering/mixing operations are performed in the audio callback thread, which has the highest priority in the system and should preempt any other running threads. Since the primary FDL has the same period as the callback, we have the option of running it in the callback thread to avoid unnecessary context switches.

In order to get the worker threads to respect the real-time deadlines of one another, we use a fixed-priority scheduling policy with higher priorities assigned to FDLs with shorter periods. On Linux, we use the “`SCHED_FIFO`” real-time policy with a max priority of 99, and on Mac, we use the “`precedence`” policy with a max priority of 63. We avoid using OS X’s “`time constraint`” policy (except when running the first FDL in a separate thread from the callback) because, even though this is the highest-priority policy recommended for real-time performance, the way that it schedules multiple threads is unpredictable, and using the precedence policy yielded much more stable audio output.

3.3. Thread Synchronization

Thread synchronization mechanisms typically rely on operating system calls, which can adversely affect real-time performance due to the variability in their execution times. Because of this, optimizing the synchronization between threads yielded the most significant performance improvements to our preemptive implementation.

We use two basic synchronization tasks in this implementation. In the first sync task, the main thread sends signals to worker threads telling them when to start. In the second, the worker threads signal the main thread when they are done. To implement these operations, we use condition variables (`condvars`) and mutex locks from the `pthread`s library. Condition variables provide a mechanism for a thread to sleep until it receives a signal from another thread, and mutexes enable a thread to lock a shared memory region to prevent other threads from simultaneously accessing it. Signaling and waiting on a `condvar` require system calls, as does locking and unlocking a mutex, so we would like to minimize our use of these routines.

In a naive approach to these sync tasks, each of the worker threads would have its own `condvar` to wait on, and the main thread would have its own set of `condvars` to wait on (one for each worker thread). The main thread would signal each of the worker threads that need to be started during the current callback via their `condvars`. Then the main thread would wait on the `condvars` that correspond to the worker threads that have deadlines during the current callback. Worker threads communicate their completion by signaling the corresponding `condvar` belonging to the main thread.

The problem with this approach is that if we have T worker threads, the main thread may need to send up to T `condvar` signals or wait on up to T `condvars` during a single callback. Using system traces, we measured these `condvar` operations to take between 3 μ s and 40 μ s (not including actual waiting time), so a few `condvar` ops

could fill a significant portion of the callback period, leaving no time for actual computation or audio I/O.

In order to reduce the number of condvars required, we can re-organize the synchronization so that all worker threads that need to be started during a single callback are waiting on a single, specific condvar. Then only a single broadcast condvar signal is needed from the main thread to start the workers. Likewise, the main thread can wait on a single condvar that is signaled by the worker thread that is last to finish. The problem with this approach is that in order for the worker threads to keep track of which is last to finish, they must all access a shared counter. If we protect this counter with a mutex, lock contention between the threads is introduced which can significantly stall their completion.

To remedy this lock contention problem, we can use atomic operations (we use the gcc builtin routines [10]), which eliminates all system calls caused by lock contention. If the worker threads atomically increment the shared counter when they complete, changing the value of the counter and getting its new value appear to occur instantaneously, negating the need for locks. This guarantees that the last thread to increment the counter will see the target counter value and know that it should send the completion signal to the condvar of the main thread. In addition, using atomic ops here allows the main thread to simply check if the counter has reached its target without having to acquire a mutex and before waiting on a condvar.

These synchronization methods allow us to get close to 100% CPU utilization on a single core without audio dropouts. Figure 6 shows how this approach works on a machine with 3 processor cores for a partitioning that uses 3 FDLs with the first FDL executing in the main callback thread. The period of the second FDL is twice that of the callback, and the third FDL has a period four times that of the callback.

3.4. Processing Multiple Channels

If we simply duplicated the scheduling and synchronization techniques outlined above for every channel when processing multiple channels, we would end up with a lot of redundant synchronization and probably a lot more threads than processor cores. To avoid this, FDLs belonging to different channels but with the same block size can be run in the same thread, since these FDLs all have the same arrivals and deadlines. Then the synchronization operations are shared amongst the channels, there is no extra synchronization overhead, and we keep the thread count low.

3.5. Targeting Multi-Core Architectures

The methods described above work fine when running on a single processor, because the thread priority assignments discussed in Section 3.2 will grant the processor to the thread with the most imminent deadline. When we have more than one processor core to work with, we can decide which core each thread should run on. Normally, the operating system will decide this for us, but this can yield suboptimal performance.

In Linux, we have the option of pinning threads to specific cores using non-portable (NP) extensions to the POSIX threads API. If we have at least as many cores as FDL levels, we could pin each FDL thread to its own core. This should minimize the number of preemptions and context switches. If we don't have enough cores to do this, we could still achieve a significant reduction in context switching by distributing the FDL threads evenly across the cores.

When processing multiple channels, another approach would be to create multiple worker threads per FDL level. This allows us to put the work belonging to a subset of the channels on each core, which would yield better load balancing across cores and possibly better memory locality. Because we would still be running all FDL levels on each core, there would still be lots of context switching, as in the single core case. We report on the performance of these thread pinning approaches in Section 5.

3.6. Choosing the Partitioning

After making all the low-level computational and scheduling decisions, we still have to decide how we will partition our impulse response(s). The approach in the Gardner paper is to double the block size every two partitions[1], but this approach fails to take advantage of FFT reuse and linearity within FDLs. Garcia has proposed a dynamic programming algorithm that determines an optimal partitioning in terms of number of mathematical operations[4]; however, this method fails to take into account actual execution time on the target system. The actual execution times of FFTs and Cmad routines operating on variably sized arrays can vary widely across hardware architectures and software implementations. This is why we feel it is important to measure the actual performance of the FDL work we are doing when choosing a partitioning, not unlike FFTWs "auto-tuning" stage.

Since we have real-time constraints, we must consider the worst-case performance of each FDL. In order to estimate the worst-case performance of an FDL of a certain block size and number of partitions, we pollute the L2 cache of our target machine prior to each execution of the FDL. The maximum-observed execution time then becomes our worst-case estimate.

To determine a best partitioning from these performance numbers, we search for the valid combination of FDLs that has the lowest overall worst-case processor load. The processor load of each FDL is calculated by dividing its maximum-observed execution time by its period. We found that it was unnecessary to search the entire FDL space since – for a specific block size – the search would always choose the minimum number of partitions required by the block size of the subsequent larger FDL; therefore, the number of partitions in each FDL was restricted to powers of two.

4. TIME-DISTRIBUTED IMPLEMENTATION

An alternative implementation strategy for non-uniform partitioned convolution is to perform all the necessary computation within a single thread, manually partitioning the work such that the workload is spread as evenly as possible across processing frames. This requires that during each frame, in addition to doing the processing for the smallest FDL, we also perform a fraction of the processing for each larger FDL. Implementing this approach required significantly more programmer effort and a deeper understanding of the underlying mathematics than the previously described preemptive approach, since we cannot rely on existing external libraries (e.g. FFTW) to perform all of the computational "heavy lifting." Although our time-distributed implementation isn't as flexible as the preemptive version (it only supports partitionings with two FDLs), it has the benefit of fitting the existing model of plugins executing within an audio host application, where a plugin is expected to do its real-time processing within the context of a single high priority thread. Currently none of the audio host applications we are

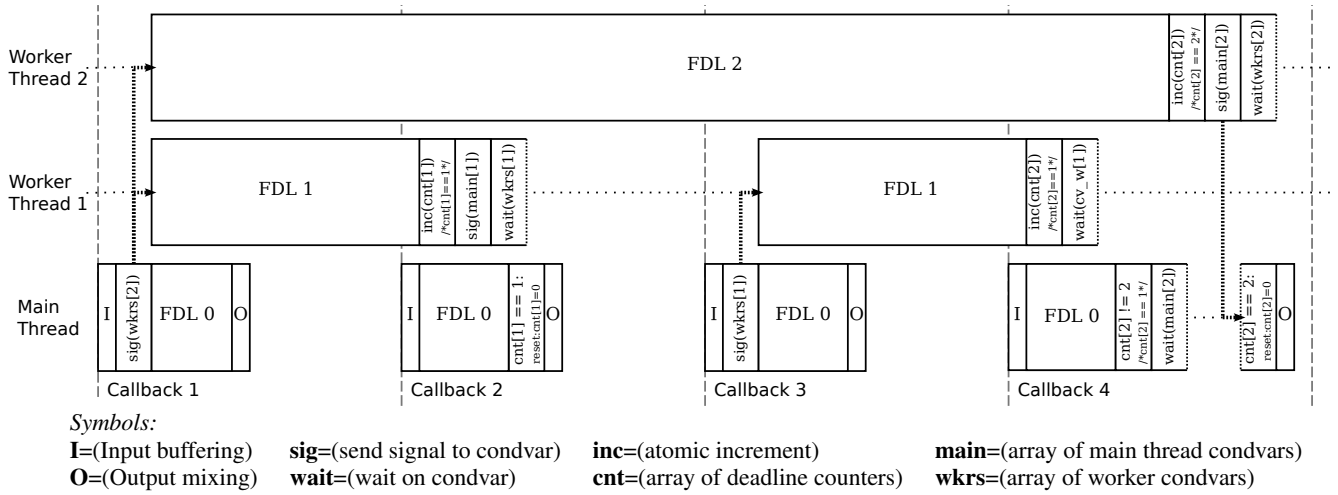


Figure 6: Preemptive-version synchronization walkthrough for a 3-FDL partitioning running on 3 cores.

aware of provide mechanisms for plugins to create and schedule the execution of additional high priority threads.

Our time-distributed implementation utilizes a technique described by Hurchalla in [2] to perform the work associated with a two level partitioning within the context of a single thread in a load balanced manner. Hurchalla describes how to apply radix-2 and radix-4 decimation in frequency (DIF), possibly in a nested fashion, to distribute the computation of a single channel of non-uniform partitioned convolution with two FDLs (where the secondary FDL is 4, 8, or 16 \times the size of the primary FDL) relatively evenly across multiple frames. DIF decomposes an input sequence into multiple subsequences which have the property that their FFT coefficients are a subset of the FFT coefficients of the input sequence. FFT-based block convolution involves three steps: calculating the forward FFT of an input sequence, performing a complex multiplication of the resulting FFT coefficients with those of a stored impulse response, and finally computing an inverse FFT to produce an output sequence. By applying DIF to an input sequence and performing the three aforementioned steps on the resulting subsequences during multiple frames, it is possible to distribute the calculations for the secondary FDL across multiple callbacks.

When using only a single stage of DIF, the work can be distributed across frames as shown in Figure 7(a). In this example, a single stage of radix-2 DIF is used to distribute the work associated with a secondary partition that is 4 \times the size of the primary partition across 4 frames.

During frames 1–4, incoming samples are buffered and a radix-2 DIF is applied to transform the input sequence A_{in} into the two subsequences A_1 and A_2 . During frame 5, the FFT of subsequence A_1 is calculated and half of the resulting FFT coefficients are multiplied with those of the impulse response. The second half of the complex multiplications are performed during frame 6, after which the IFFT of the resulting coefficients is computed. The same operations are performed on the subsequence A_2 during frames 7 and 8. Finally, an inverse DIF is applied to the two subsequences computed during frames 9–12 and the resulting real sequence is the output sequence A_{out} . In this example, the workload is perfectly balanced across frames since the same amount of work is performed for each of the 3 steps (input, intermediate, output) that

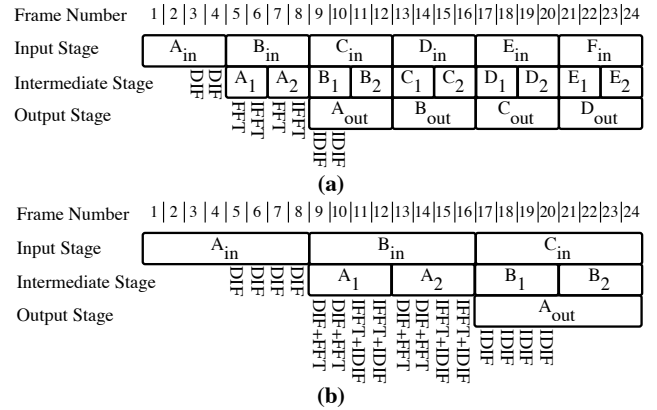


Figure 7: Time-distributed processing walkthrough. (a) Secondary partition 4 \times primary partition (b) 8 \times primary partition

are performed during each frame.

Figure 7(b) illustrates one way to distribute the work for a secondary partition that is 8 \times the size of the primary partition by using two nested stages of radix-2 DIF. During frames 1–8, input samples are buffered and DIF is applied as in the previous example, but the work is spread out over twice as many frames. During frame 9, the first part of a radix-2 DIF is applied to subsequence A_1 to generate a new subsequence A_{11} . The FFT of this sequence is computed, and half of the complex multiplications of the resulting FFT coefficients with the impulse response FFT coefficients are performed. The second part of the radix-2 DIF is computed during frame 10 to produce the subsequence A_{12} . The FFT and half of the complex multiplications are performed, as for subsequence A_{11} during the previous frame. During frame 11, the second half of the complex multiplications started in frame 9 are completed, the IFFT is computed and half of the inverse DIF is performed. Finally during frame 12 the complex multiplications started in frame 10 are finished and the second portion of the inverse DIF calculation is concluded, resulting in a complete output sequence corresponding to the input sequence A_1 . The same computations are performed

on the sequence A_2 during frames 13–16, and the resulting output sequence is combined with the one from frames 9–12 to produce the output sequence A_{out} during frames 17–24. Unlike the previous example, in this case the workload isn’t perfectly balanced because the work associated with the nested forward and inverse DIF steps varies somewhat across frames. For more details, see [2].

The time-distributed partitioned convolution implementation we evaluate in Section 5 uses two stages of radix-4 DIF to decompose the input sequence to the secondary FDL into 16 subsequences. During each frame of processing, we take the FFT of one of the input subsequences and perform half of the complex multiplications with the impulse response FFT coefficients. During a subsequent frame, we perform the second half of the complex multiplications and take the inverse FFT of the resulting values. This enables us to distribute the FFT, complex multiplication, and inverse FFT steps relatively evenly across 32 processing frames. We use FFTW to perform the “leaf-level” FFT calculations. During each frame we also do a portion of the forward and inverse DIF decomposition for the previous block of input and the current block of output. This processing is not perfectly distributed across frames, resulting in a slight imbalance of the work done from frame to frame. The measured variation in execution time across frames for this implementation is less than 5 percent.

In [3], Hurchalla describes a method for applying nested short-length acyclic convolution algorithms to improve the computational efficiency of the complex arithmetic performed in the frequency domain. The basic idea is to treat each frequency bin in each partition of the impulse response as a sequence, and to perform a running convolution between this sequence and the corresponding frequency bin of the FFT of the input signal. We implemented a basic version of Hurchalla’s scheme, using a single stage of 3-partition acyclic convolution. These convolution routines, as well as the routines used to perform the forward and inverse radix-4 decomposition steps, were hand optimized in assembly using the SSE extensions to the x86 ISA. While this scheme did reduce the overall amount of work done (in terms of the total number of floating point operations executed), we found that the variation in execution time from frame to frame was greater than when using a naive implementation of convolution. This resulted in a longer worst case execution time, which meant that the version of the code that included the optimized convolution routines was never able to concurrently process as many independent channels of convolution as the version using the naive convolution routines. For this reason, we do not include an evaluation of the code using this optimization in the following section. Hurchalla also discusses various techniques to time distribute work across multiple frames when working with multiple channels. We did not implement any of these techniques – when operating with multiple channels, our implementation processes each channel independently.

5. EXPERIMENTAL RESULTS

In this section we present and analyze performance measurements of our preemptive and time-distributed implementations of partitioned convolution. The machine used to perform the benchmarking was a Mac Pro with two 2.66 GHz 6-core Intel Xeon “Westmere” processors and 12GB of memory, running Linux 2.6.35 with low-latency realtime patches applied. We only enabled one of the two sockets and disabled Hyperthreading for all the experiments described in this section. A 10-channel ethernet audio

interface[11] was used for I/O. This audio device behaves similarly to a Firewire or USB based device but uses Ethernet as its transport. All experiments were performed at a sample rate of 44.1 kHz using a 64 sample frame size. The impulse response lengths we considered range from 16,384–524,288 samples (0.4–11.9 seconds). To make our results as deterministic as possible, we disabled all frequency scaling mechanisms present in the operating system, as well as Turbo Boost (hardware based opportunistic frequency scaling) in the CPU.

Each implementation was written as a standalone application that takes arguments specifying the number of channels (instances) of convolution to perform and the impulse responses to use. For the preemptive implementation the partitioning must be specified whereas for the time-distributed version it is fixed (two partitions, the second being $32\times$ the size of the first). All implementations interface with the audio subsystem using the ALSA API (the Linux audio interface standard) directly – as opposed to using a cross-platform library (such as PortAudio) or daemon (such as JACK) – in order to minimize overhead. Communication between the application and OS is done through shared regions of memory mapped into the application’s address space, and the OS notifies the application that a new frame of audio is ready by executing a callback function asynchronously in a high priority thread.

5.1. Single-Core Measurements

For our first experiment, we disabled all but a single core in the system and recorded the reported CPU utilization (averaged over one second) for three different configurations executing the same workload. The workload was 16 independent channels of partitioned convolution, and the three configurations were: preemptive using two FDLs, preemptive using the empirically derived optimal partitioning (ranging from 3–5 FDLs), and time-distributed. The results are presented in Figure 8.

The time-distributed and preemptive two-level implementations use the same partitioning so we might expect them to exhibit a similar computational load. This is true for the smaller partition sizes, but for larger partition sizes with more computationally intensive workloads, the time-distributed implementation appears to have a clear advantage. We hypothesize that this is due to the overhead of context switches and system calls needed for preemptive and synchronization in the preemptive version. Each context switch or system call requires a trap into supervisor mode and the operating system kernel which incurs significant overhead. The preemptive version using the optimal partitioning scheme outperforms all others by a wide margin.

Our second experiment was to measure how many instances (independent channels of convolution) each implementation was capable of running without experiencing any missed deadlines or dropouts. This experiment was also performed using only a single core. We increased the number of instances until we reached the highest point that ran dropout-free for 60 seconds. Figure 9 illustrates the results, which are quite different from what the results of the previous experiment would suggest. In this case, the two-level preemptive implementation is able to achieve more concurrent instances without dropouts than the time-distributed version. We believe this is due to several factors: the imperfect load balancing of the time-distributed version, the greater regularity and predictability of the memory access patterns in the preemptive version, and the reduced sensitivity to the timing of callback function arrivals in the preemptive version. The previous graph reported

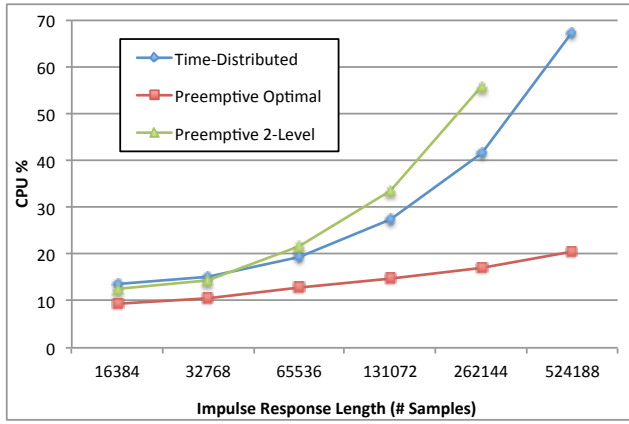


Figure 8: CPU utilization for a single core performing 16 channels of convolution.

average CPU load over many frames, but what is important in determining the maximum number of sustainable instances without missed deadlines is the worst-case execution time (WCET) during any individual frame. For the time distributed version, the WCET is higher than the average execution time. Whereas the time distributed version only performs a portion of the computation related to the secondary FDL during each frame, the worker threads of the preemptive version process higher-level FDLs to completion (unless they are preempted). This results in long streams of memory accesses with a constant stride, and the code is therefore able to benefit from the hardware prefetching mechanisms in the memory hierarchy to reduce the latencies caused by cache misses. The time distributed version is also more sensitive to variations in execution time of the callback function, since it must complete all of its work before the arrival of the next callback. The preemptive version only has to complete a fraction of the total work associated with the convolution during the callback since much of the work is being done in different threads. This means that it is better able to tolerate jitter in the arrival times of the callback functions. Once again, the preemptive implementation using the optimal partitioning scheme is the clear winner, outperforming the others by a factor of $4\times$ for the longest impulse response.

5.2. Multi-Core Measurements

Our final experiment was to benchmark the preemptive implementation running on multiple cores. We assigned a single thread to process each FDL and pinned each thread to its own core to minimize disturbances from the OS scheduler. Any cores that weren't necessary for a given experiment were disabled. Since the optimal number of FDLs varies with impulse response length, so do the numbers of cores we used in these experiments. As mentioned in Section 3.5, we also considered an alternative scheme where channels (instead of FDLs) were distributed amongst the cores. In this case, one thread per FDL level was pinned to each core – so for N FDLs and M cores there would be a total of $N \times M$ threads active in the system. However, the pin-by-FDL scheme outperformed the pin-by-channel scheme in all measurements, so we only present the results from the former here.

A plot comparing the performance of the code running on single and multiple-core configurations is presented in Figure 10. By

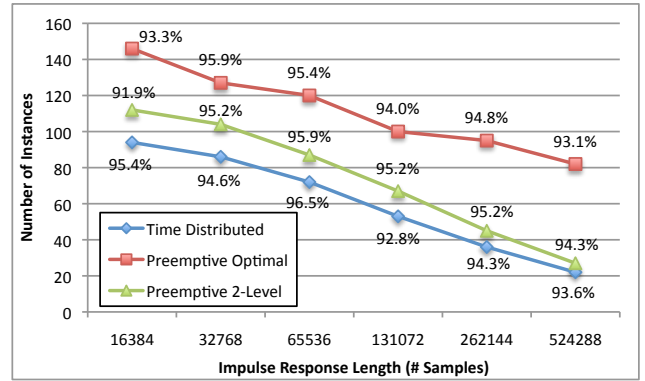


Figure 9: Maximum number of independent channels of convolution possible without dropouts for various implementations running on a single core. Points are labeled with reported CPU utilization.

using additional cores, we were able to run between $1.3\times$ and $1.7\times$ as many instances without experiencing dropouts. While our work partitioning scheme is most likely not optimal (there is significant variation in the computational load across FDLs), we believe the factor that ultimately limits the maximum achievable number of independent instances is memory bandwidth, not computational crunch. The processor used for these experiments has 12MB of last level cache and 256KB of private level 2 cache per core. A 524,288 sample impulse response represented as single precision floating point values occupies 2MB of memory. Clearly for the large number of concurrent instances we are able to run, the working set doesn't fit into the on-chip cache and the latency of DRAM accesses becomes a bottleneck for achievable performance.

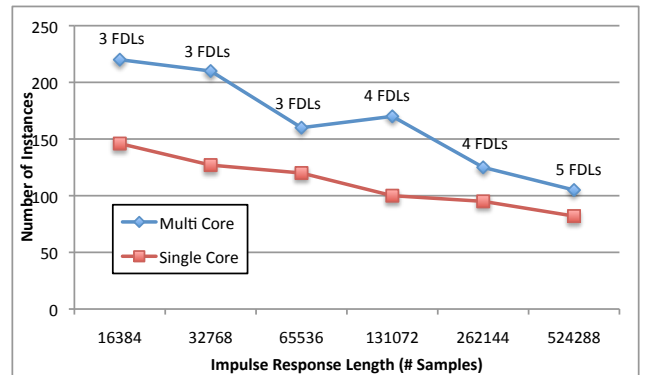


Figure 10: Maximum number of independent channels of convolution possible without dropouts for single and multi-core cases (preemptive implementation). Points are labeled with the number of FDLs used.

6. DISCUSSION

In all of the scenarios we investigated, the preemptive implementation of partitioned convolution, using an empirically determined optimal partitioning, outperformed all of the others by a wide margin. Our motivation for implementing the time-distributed version was to be able to use it in the context of an audio processing environment such as Max/MSP[12] or Pd[13]. However, this is just a

stop-gap solution, and in the future we hope that audio host applications will provide mechanisms for plugins or objects to schedule their execution across multiple concurrent threads.

Another advantage of the preemptive approach lies in the programmer effort required to implement it. While efficiently managing the scheduling of multiple threads is not trivial, it affords us the opportunity to use existing highly optimized libraries to perform necessary computations without needing to worry about manually partitioning the work. Optimizing the time-distributed FFT to the point that it was competitive with FFTW's FFT routines required hand tuning assembly code and carefully managing data layout which was an arduous task. Also, the techniques used to implement the time-distributed FFT don't scale well to larger (greater than 32x) FDL partition sizes, which limits the performance of the time-distributed partitioned convolution algorithm for very long impulse response lengths.

6.1. Further Optimizations

Despite the fact that our time-distributed implementation performed worse in nearly every aspect, this approach still has room for improvement – though the obvious improvements, such as taking advantage of the regularity across channels to more optimally distribute the computation [2], would take significant programmer effort and restrict the implementation to specific use cases.

For the preemptive version running FDLs on separate cores, the computational load on each core is not evenly balanced. We could attempt to balance the load on each core during our optimal partitioning search; however, this is only likely to yield a slight improvement due to the fact that none of the CPUs approach 100% utilization when dropouts begin to occur. This points to the fact that the implementation, in its current state, is limited by the memory bandwidth between the cores as mentioned at the end of Section 5; therefore, we feel that further multi-core optimizations would be best geared toward reducing memory traffic and optimizing cache usage amongst the FDLs.

6.2. The Need to Support Preemption and Multi-threading

Partitioned convolution is but one example of a class of multi-rate audio processing and analysis tasks, others include score following, rhythm and pitch extraction, and algorithmic composition. Generally speaking, it can be quite cumbersome (if not impossible) for the programmer to time-distribute long-running tasks evenly across multiple short time periods, particularly when those tasks call external libraries. For this particular case (FFTs) there are clever tricks that allowed us to accomplish this in a limited manner but other computations (for example, those related to machine learning algorithms) may not be as amenable to such treatment.

While it is possible for us to spawn worker threads and attempt to manage them, even while running in the context of existing audio host applications, there is no guarantee that other plugins running on the host won't do the same thing. This would result in pollution of our "thread ecosystem" and force our threads to compete with others for processor time and cache space. Ultimately, when there are more threads than cores in the system, the responsibility for scheduling the threads falls onto the operating system, which can only do so well given that it has very limited knowledge about the relationships and dependencies between threads.

If the audio host itself were able to manage its hardware resources (processor cores and caches) by arranging the execution

of plugin tasks on multiple cores, then plugin programmers and end users could benefit from improved parallel performance, programmability, and quality of service. On-going research into the development of an operating system that enables applications with real-time constraints running on multi-core machines to more explicitly schedule and "micro-manage" the execution of their constituent threads is described in [14].

Acknowledgments

This work was supported in part by Microsoft (Award #024263) and Intel (Award #024894, equipment donations) funding and by matching funding from U.C. Discovery (Award #DIG07-10227).

7. REFERENCES

- [1] WG Gardner, "Efficient convolution without input-output delay," *JAES*, vol. 43, no. 3, pp. 127–136, 1995.
- [2] J Hurchalla, "A time distributed FFT for efficient low latency convolution," *Audio Engineering Society Convention 129*, 2010.
- [3] J Hurchalla, "Low latency convolution in one dimension via two dimensional convolutions: An intuitive approach," *Audio Engineering Society Convention 125*, 2008.
- [4] G Garcia, "Optimal filter partition for efficient convolution with short input/output delay," *113th Convention of the Audio Engineering Society*, 2002.
- [5] A. Oppenheim and R. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, Englewood Cliffs, 1999.
- [6] D Orozco, L Xue, M Bolat, X Li, and G.R Gao, "Experience of optimizing FFT on Intel architectures," *2007 IEEE International Parallel and Distributed Processing Symposium*, p. 448, 2007.
- [7] Matteo Frigo and Steven G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.
- [8] Agner Fog, "Instructions for asmlib: a multi-platform library of highly optimized functions for C and C++," <http://www.agner.org/optimize/asmlib-instructions.pdf>, 2010.
- [9] Bradford Nichols, *Pthreads Programming: a Posix Standard for Better Multiprocessing*, O'Reilly, Sebastopol, 1996.
- [10] Using the GNU Compiler Collection (GCC), "Built-in functions for atomic memory access," <http://gcc.gnu.org/onlinedocs/gcc/Atomic-Builtins.html>, 2011.
- [11] Rimas Avizienis, Adrian Freed, Takahiko Suzuki, and David Wessel, "Scalable connectivity processor for computer music performance systems," in *Proc. of the Int'l Computer Music Conference*, Berlin, Germany, 2000, pp. 523–526, International Computer Music Association.
- [12] Cycling'74, "Max/MSP," <http://cycling74.com>.
- [13] Miller Puckette et al., "Pd," <http://puredata.info>.
- [14] Juan Colmenares et al., "Real-time musical applications on an experimental operating system for multi-core processors," in *Proc. of the International Computer Music Conference*, Huddersfield, England, 2011.

TRANSFORMING VIBRATO EXTENT IN MONOPHONIC SOUNDS

A. Roebel

Analysis/Synthesis team
STMS - IRCAM CNRS UPMC
Paris, France

`axel(dot)roebel@ircam(dot)fr`

S. Maller,*

Analysis/Synthesis team
STMS - IRCAM CNRS UPMC
Paris, France

`simon.maller@audionamix(dot)com`

J. Contreras

Analysis/Synthesis team
STMS - IRCAM CNRS UPMC
Paris, France

ABSTRACT

This paper describes research into signal transformation operators allowing to modify the vibrato extent in recorded sound signals. A number of operators are proposed that deal with the problem taking into account different levels of complexity. The experimental validation shows that the operators are effective in removing existing vibrato in real world recordings at least for the idealized case of long notes and with properly segmented vibrato sections. It shows as well that for instruments with significant noise level (flute) independent treatment of noise and harmonic signal components is required.

1. INTRODUCTION

Signal parameter transformation is one of the main interests in professional audio applications. Today there exist numerous methods allowing the independent transformation of pitch, spectral envelope, and duration achieving in many cases signal quality that is not distinguishable from a real recording, and accordingly new possibilities can be explored. One of the potentially very interesting objectives is expressive signal transformation. Under this term we will understand a dynamic signal transformation that has the objective to add or change the perceived affective state of the sound. Expressive speech signal synthesis has become an increasingly active research area [1, 2, 3, 4, 5] and expressive music signal synthesis has already become a reality for instrument sound signal synthesis [6]¹. Compared to the expressive sound synthesis the topic of expressivity transformation is significantly more complex because we do not

have as much control and information about the sound signal to be transformed, and moreover, we may have to remove an existing expressive parameter variation to be able to introduce the desired expressivity. For expressive speech transformation there exist only rather few approaches [7, 8] that generally start with a neutral input signal.

In the present paper we will investigate into the problem of manipulating vibrato, which one of the important expressive means for the case of musical input signals. There exist quite a few studies that are related to the manipulation of vibrato [9, 10, 11, 12, 13, 14, 15]. A common starting point for all these studies is the estimation of the fundamental frequency, which is then followed by a separation into a slowly evolving fundamental frequency contour and the fast ornamentation contour representing the vibrato. The approaches then diverge and establish different control strategies for the adaption of the partial amplitudes and frequencies. The objective of vibrato manipulation can be diverse. The most ambitious case requires independent manipulation of the vibrato parameters *extent* (vibrato amplitude), *rate* (modulation frequency) and *form* (contour). While the order-2 sinusoidal model introduced in [13] provides theoretical means to control the vibrato form, to the best of our knowledge, this has not yet been considered in a practical implementation. Vibrato rate manipulation can be obtained by means of pitch shifting the modulated parameter contours using again either the order-2 sinusoidal model, or by means of the analytic signal of the modulated parameter [11], or by means of using time domain OLA techniques combined with resampling.

In the present study we limit ourselves to the problem of the modification of the vibrato extent. The basic idea of the method is conceptually simple. It consists of applying the separation into slowly varying contour components and modulated ornamentation not only to the fundamental

* Now with Audionamix, Paris

¹This side by side presentation of expressive speech and music synthesis does not intend to imply that expressive effects will be generated by means of similar means for these two signal classes.

frequency but to all other sound parameters as well. The vibrato control manipulation will then become a simple mixing of the modulated parameter component. This strategy can in principle be applied to sinusoidal partial parameters (amplitude and frequency trajectories) as well as to the complete spectral envelope of the modulated sound. The spectral envelope based method allows to unify the treatment of the sinusoidal and noise modulations and therefore it will be used in the following discussion. When compared to methods using explicit sinusoidal models as for example [13, 14] the proposed method is simpler yet rather efficient. We note that nearly all existing studies focus entirely on the effect of the vibrato on the sinusoidal components. There exist only very few studies dealing with the effect of the vibrato on the noise components of the sound [16]. The present experimental investigation shows that the modulation of sinusoidal and noise components that is induced by vibrato is not synchronous. Therefore, the effect of the vibrato on the noise component cannot be neglected when a perceptually convincing removal of the vibrato is desired.

For the experimental investigation we have selected to use the problem of removing vibrato from a monophonic music instrument recording. The idea is that removing signal modulations is perceptively the most critical operation because any residual modulation will be easily perceived. Due to the fact that the presence or absence of modulation can be judged rather easily even when only a low level of modulation is present the problem allows for relatively simple perceptual evaluation. Please note as well, that the investigation is only concerned with the transformation of the extent of the vibrato ornamentation. Many other problems exist that should be addressed. Examples are the problems related to the detection of vibrato [17, 18], the change of the vibrato rate when time stretching a signal, the coherent transformation of note transitions, and the generation of concrete pitch and spectral envelope modulation contours if vibrato should be added to an unmodulated signal.

The organization of the article is as follows. In section 2 the model of vibrato generation is presented and discussed. Based on this model section 3 introduces dynamic signal transformation operators that allow to remove the effects related to vibrato with different levels of refinement. In section 4 these operators are experimentally evaluated using a real world flute signal and section 5 describes the conclusions that can be drawn from the investigation.

2. SIGNAL MODEL FOR VIBRATO ORNAMENTATION

Vibrato is an expressive ornamentation that is frequently used in music [19, 14]. The term vibrato is generally understood to refer to a quasi periodic modulation of the pitch or fundamental frequency. But, as will become clear in the fol-

lowing, pitch modulation without energy and timbre modulation does not exist in real world sound sources. Therefore, a realistic manipulation of vibrato requires a coherent transformation of timbre and energy modulations as well. We have selected vibrato for our experimentation with expressive signal transformation because: first it is widely used, second it includes effects like energy and timbre modulation and third, due to its periodic nature it is perceptually much easier to evaluate to access than for example note transitions. Transformation of expressive note transitions are currently under investigation and will not be addressed in the present study.

Vibrato is generally produced by means of a periodic variation of the fundamental frequency of the musical note. Accordingly, the vibrato signal has a quasi-periodic variation of the fundamental frequency around a central value. Perceptually, this modulation does not directly affect the perceived pitch of the note [19]. The perceived pitch of vibrato notes has been investigated repeatedly and has been found to be given by the a value close to the mean pitch over a vibrato period [20]. For short vibrato of less than 2 vibrato periods the perceived pitch is affected more by the end of the pitch evolution [21]

Different signal models including pitch variation with constant spectral envelope, pitch and amplitude modulation as well as pitch, amplitude and spectral envelope modulation have been compared in [14, 15]. In a natural physical musical instrument (including singing voice) the modification of the fundamental frequency will necessarily be accompanied by amplitude and timbre modulation. On one hand the partials are moving continuously through the different resonances of the sound source resonator which will generate an induced timbre modulation. The change of the pitch will moreover require a reconfiguration of the physical configuration of the excitation and the resonating parts leading to a more or less pronounced timbre modulation. Perceptual experiments have shown that the timbre modulation that comes with the vibrato modulation is perceptually more important than the fundamental frequency modulation itself [12] and that vibrato without independent timbre modulation does not sound natural and physically plausible [14]. An interesting question that arises is related to the modulation of the noise components of the signal. Noise components do not necessarily take the same acoustic path and therefore, we assume in the following that the modulation of the noise components will not be in sync with the modulation of the sinusoidal components.

Based on the results discussed so far we will establish our vibrato transformation algorithm taking into account pitch as well as amplitude and timbre modulation. For each partial we assume a amplitude trajectory that is given by

$$A_k(n) = \exp(S(w_k(n), n)), \quad (1)$$

where A_k represents the time varying amplitude of partial k , n is the discrete time, $S(w, n)$ is a quasi-periodically time-varying spectral envelope obtained from the log amplitude spectrum that modulates due to the physical reconfiguration of the instrument and w_k is the quasi-periodically time-varying frequency of partial k . We note that the existing amplitude modulation and also potential amplitude estimation errors that may be related to the modulation are absorbed into the time-varying spectral envelope S . Similarly for the noise component we assume a time varying distribution of the log amplitude spectrum given by $N(w, n)$ that may have a periodic structure in n .

3. VIBRATO TRANSFORMATION

The objective of the present section is to develop an algorithm that allows continuously controlling the vibrato extent. This control is performed for the part of the note that is located between attack and release and the region of attack and release are assumed to be known. In the following experiments the attack and release segments have been obtained by means of manual labeling.

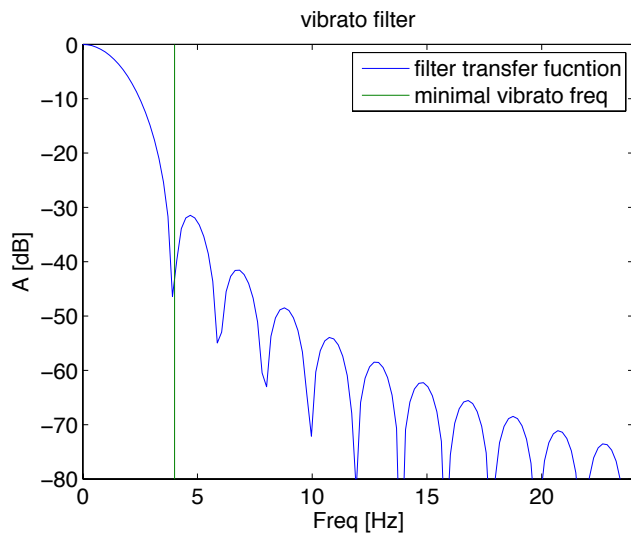


Figure 1: Vibrato filter transfer function for a vibrato filter assuming minimum vibrato frequency of 4Hz and 2 filter length of 2 times the maximal vibrato period.

To achieve this control the time-varying fundamental frequency and the time varying spectral envelope S or N will be represented by means of a superposition of a constant and a quasi periodic part. The constant part is derived by means of smoothing the time varying evolution of $w(n)$ and $S(w, n)$ over the local period of the fundamental and the quasi periodic part is then the difference between the original function and the smoothed version. Accordingly, the vibrato control can be achieved by means of syn-

chronously scaling the dynamic part of the fundamental frequency and the spectral envelope as follows

$$w_k(n) = \bar{w}_k(n) + \alpha \tilde{w}_k(n) \quad (2)$$

$$S(w, n) = \bar{S}(w, n) + \alpha \tilde{S}(w, n). \quad (3)$$

$$N(w, n) = \bar{N}(w, n) + \alpha \tilde{N}(w, n). \quad (4)$$

Here \bar{w} , \bar{S} and \bar{N} are low pass filtered versions of w , $S(w, n)$, and $N(w, n)$ and \tilde{w} , \tilde{S} , and \tilde{N} are the respective high-pass residuals that represent the modulated part of the related parameter. α is the control parameter that can be used to scale the modulation. Note, that the spectral envelopes are using log amplitude representation such that the variation is scaled geometrically. When compared to existing approaches for manipulation of the spectral envelop modulation the proposed scaling operator has a number of advantages. We discuss 2 examples. [9] proposed to find the target contour of the transformed envelope by means of interpolating between two waveform tables. The two tables were obtained at the time positions related to the extreme values of the fundamental frequency that were determined for each vibrato period. As interpolation control variable they used the instantaneous fundamental frequency. [15] uses a similar principle interpolating two spectral envelopes obtained from the extreme positions of the fundamental frequency. They proposed to control the linear interpolation by means of a sinusoidal interpolation contour. Both approaches suffer from 2 problems. First, they make a hypothesis about the form of the fundamental frequency contour, notably that there exist two robustly identifiable positions that can be used to anchor the interpolation. Second, and more importantly, they cannot represent the original signal. Even if no signal transformation is desired the model would still not create the unmodified spectral envelope. In contrast to this, the transformation proposed in eq. (3) will always be a neutral operation for $\alpha = 0$ and second it will preserve the local form and evolution of the envelope modulation even for $\alpha \neq 0$ as well as the phase relations between the modulations of the fundamental and all the partial amplitudes.

In the following we will assume that the sound sources are quasi harmonic such that the scaling of the frequency trajectories can be obtained by means of dynamic transposition. Accordingly, instead of operating the frequency trajectories of the individual partials only the fundamental frequency trajectory $w_0(n)$ needs to be treated².

3.1. Extracting modulated parameter contours

To achieve the separation between the two parameter contours that are required for each parameter to establish eq. (3) we propose to use a FIR filter that is given by a standard window function having at least 2 times the length of

² Note however, that the same principles could be applied to the individual amplitude and frequency parameter contours of a sinusoidal model.

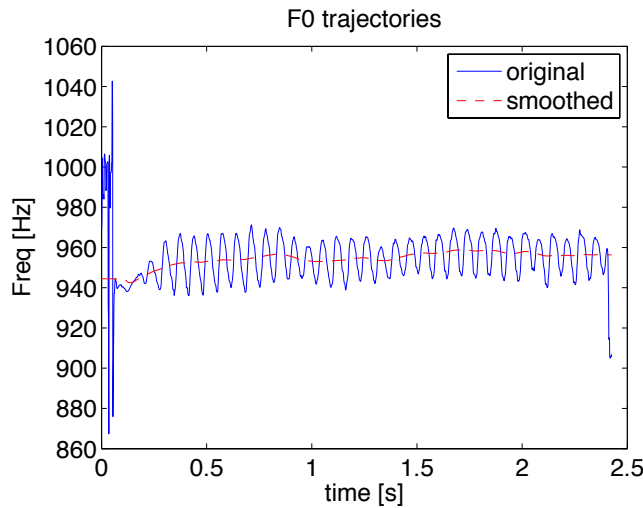


Figure 2: Result of vibrato suppression on a real world vibrato note of a flute signal using vibrato filter shown in fig. 1. See text for more explanations.

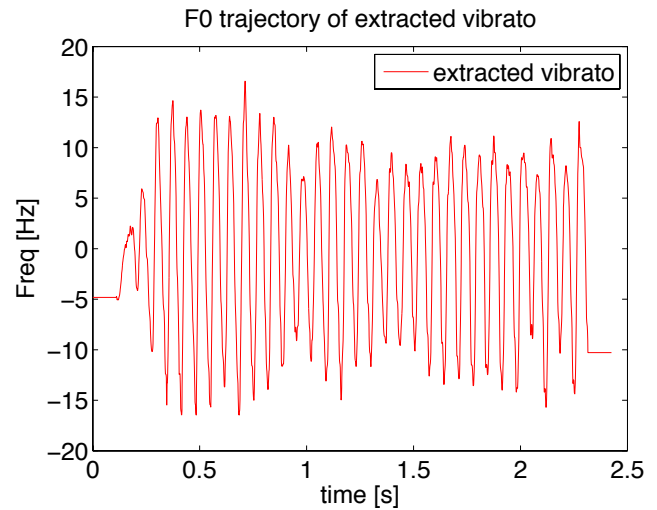


Figure 3: The vibrato residual obtained by means of subtracting the smoothed f_0 trajectory from the original trajectory shown in fig. 2. See text for more explanations.

the period of the minimal vibrato frequency. The window is normalized to have a sum of 1. The advantages of this filter specification are the facts that

- the filter has linear phase
- the transfer function can be dynamically adapted to the vibrato frequency such that the zeros of the transfer function are located at the harmonics of the vibrato frequency.

In the following we use a Hanning window, but other window functions may be used. fig. 1 shows the transfer function obtained by means of using a Hanning window of 0.5s which is 2 times the period of a minimum vibrato frequency of 4Hz. The minimum amplitude rejection above the vibrato frequency limit is 30dB. Note that fundamental frequency variations of about 3Hz are already attenuated by about 20dB.

The application of the vibrato suppression filter to a flute vibrato of about 6Hz is shown in fig. 2. In this example the limiting parts of the smoothed f_0 trajectory have been replaced by constant values for the range of values where the window is not completely inside the trajectory. The residual that results after subtracting the smoothed F_0 trajectory from the original F_0 trajectory is shown in fig. 3. The samplerate of the F_0 trajectory in the present and all following cases is 100Hz.

3.1.1. Determining the target F_0 contour

In the following section the relation between the original and smoothed fundamental frequency trajectories will be

used to derive the signal transformations that are necessary to obtain a convincing control of the vibrato extent. To achieve the control of the vibrato extent the control parameter α is used and a segmentation of the fundamental frequency sequence into segments with and without vibrato is assumed [17]. The segments with vibrato are collected into a group of segments that are accessed individually by means of index k . This parameter α is then used to obtain the target fundamental frequency for segment k as follows

$$F_{k,target}(n, \alpha) = (F_k(n) - F_k(\bar{n}))\alpha + F_k(\bar{n}), \quad (5)$$

where $F_k(n)$ is the original fundamental frequency trajectory, $F_k(\bar{n})$ its smoothed counter part and $F_{k,target}(n, \alpha)$ the fundamental frequency target trajectory. Accordingly, by means of setting $\alpha = 1$ the fundamental frequency is kept unchanged, with $\alpha = 0$ the vibrato is removed, $\alpha = -1$ the vibrato is inverted and with $\alpha = 2$ the vibrato extent can be amplified.

3.2. Signal transformation

In this section a number of vibrato transformation algorithms will be developed that provide an increasing level of refinement with respect to the effects that can be taken into account. The idea here is to experimentally evaluate the different strategies on the perceptually most demanding transformation that removes the vibrato completely ($\alpha = 0$). The different strategies will then be evaluated on a short example such that the required level of refinement for high quality vibrato control can be determined.

The transposition to be used to transform the pitch of

the sound signal is given by

$$T_k(n) = \frac{F_{k,target}(n, \alpha)}{F_k(n)} \quad (6)$$

3.2.1. PS: preservation of spectral envelope

The first level of refinement consists of the application of the dynamic transposition specified in eq. (6) under preservation of the spectral envelope. This preservation is done by means of estimating the spectral envelope with the spectral envelope estimator proposed in [22, 23]. The related transformation operator is denoted as $PS(\alpha)$. This operator allows to take into account the timbre modulation that is induced by the modulation of the fundamental frequency assuming that the spectral envelope does not change with the modulation of the fundamental frequency.

3.2.2. PSCS: correction of spectral envelope modulation

In case that the modulation of the spectral envelope is perceptually significant after vibrato modification a refinement of the previous model is needed that allows to remove the modulation of the spectral envelope. This can be achieved by means of smoothing the spectral envelope. Similar to the procedure described for calculating the transformed fundamental frequency the PSCS(α) operator calculates the target spectral envelope by means of eq. (3). The extraction of the non modulated part $\bar{S}(w, n)$ of the spectral envelope is done similar as for the F0 trajectory besides that here we use a simple rectangular window having the length of approximately the spectral vibrato period. Again the modulated part of the spectral envelope $\tilde{S}(w, n)$ is obtained as residual after subtraction $\bar{S}(w, n)$ from the original envelope. Note, that the demodulation proposed here for spectral envelope smoothing will at the same time remove amplitude modulations and timbre modulations related to local deformation of the spectral envelope.

3.2.3. PSCTRT/PSCTNRT: correction of timbre modulation with and without residual transformation

The PSCS operator allows taking into account the modulation of the spectral envelope. In the experimental investigation we found, however, that noise and harmonic components do not follow the same modulation pattern. Accordingly, the smoothing of the spectral envelope will not remove the modulation in the non harmonic parts of the spectrum. To be able to handle 2 different modulation patterns in the noise and harmonic spectrum two variants for correction of timbre modulation have been developed. Both versions, the PSCTRT(α) as well as the PSCTNRT(α) operator establish a separation of the signal into harmonic components

and residual noise. This separation of harmonic and non harmonic signal components is done with a standard harmonic sinusoidal model [24]. The harmonic component of the sound is treated with the PSCS(α) operator. The target envelope for the noise component is again calculated using eq. (3) but in the case the noise envelope is calculated using a small order AR model of the residual. In the present experiment we use an AR order of 20. The separation into modulated and non-modulated noise filter is done strictly equivalent to the method discussed above. The difference of the PSCTRT and the PSCTNRT operator is then that the former transposes the residual with the harmonic signal when the fundamental frequency modulation is removed and the latter extracts the residual before transposition such that the residual signal is not transposed. The difference is expected to be important when the residual signal contains sinusoidal components that may or may not be modulated with the fundamental frequency.

4. EXPERIMENTAL RESULTS

In this experimental evaluation the different versions of vibrato manipulation operators have been applied to different signals, e.g. singing, cello, and flute. In the following the results obtained for the flute signal will be presented because for this signal a version with and without vibrato was available. Moreover, for the flute signal a strong noise component is present such that all the different strategies for the treatment of timbre modulation can be evaluated. All signal transformations have been performed with an enhanced phase vocoder implementation that has been developed at IRCAM. Results obtained are not bound to this technology however, and many other approaches can be used to achieve the same results. The separation of sinusoidal and residual components has been performed using a harmonic sinusoidal model using a parameter estimation based on quadratically interpolated amplitude of spectral peaks [25] without any bias correction [26].

In the top row of fig. 4 the spectrogram of a short extract of two original flute signals played by the same player on the same flute with and without vibrato is shown. The pitch of both notes is approximately at 950Hz. The 4 spectrogram displayed below are extracts of the signals transformed using the 4 vibrato modification operators defined above. In the center left position the result of operator PS(0) is displayed³. The fundamental frequency modulation has been removed but all partials as well as the noise still present a clear amplitude modulation. Perceptually this modulation is rather strong leading to a very unsatisfying result. In the figure in center right position the results after additional transformation of the spectral envelope modulation by means of

³For sound examples please visit <http://anasynt.h.ircam.fr/home/english/media/dafx11-demo-vibrato-removal>

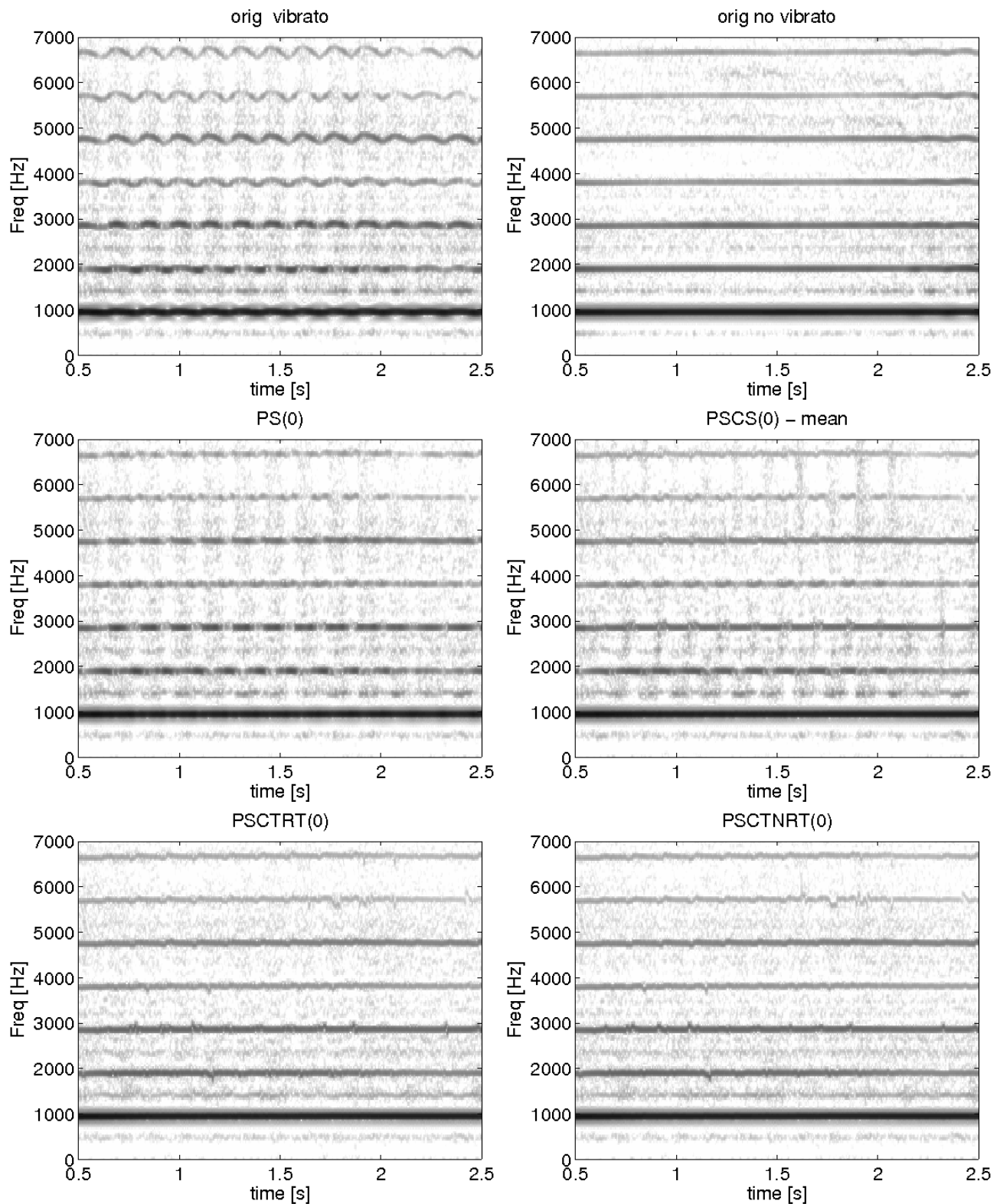


Figure 4: Experimental results for vibrato removal using the different operators described in the text. The top left figure shows a spectrogram of 2 seconds of a flute signal with vibrato, on the top right the same flute playing the same note without vibrato. The bottom 4 figures show the spectrogram of 4 different vibrato extent transformation operators introduced in the text. See there for detailed explanations.

the PSCS operator is shown. The amplitude modulation of the partials is significantly reduced. Note however, that the amplitude modulation of the noise is now significantly stronger. This is clearly visible for partials 3 and 7 that in the spectrogram of the original signal with vibrato both exhibit a strong amplitude modulation while the surrounding noise does not or hardly modulate in amplitude. After transformation the noise modulation around these 2 partials is significantly increased. Partial 5 in contrast does not suffer from this effect because partial and surrounding noise have approximately similar amplitude modulation.

The last row of fig. 4 shows the results obtained with the complex operators that separate harmonic and residual components and separately treat the spectral envelope modulation of both components. Both operators achieve a signal with a spectrogram that is very close to the flute signal with no vibrato in the top right figure. Both operators suffer from partial tracking problems in the areas where partials disappear in noise. This effect is perceptually weak, however, when carefully listening with headphones it can be perceived. It should be noted however, that it is easy to imagine a situation where this effect becomes a perceptual problem in which case a more complex transformation operator allowing for partial reconstruction would be required. The PSCSRT operator exhibits a slight modulation of the noise resonance that is located between partials 1 and 2 which is due to the fact that this operator applies the dynamic transposition that is required to remove the frequency modulation to both residual and harmonic component. In the informal listening tests with expert listeners this difference is hardly perceivable.

5. CONCLUSIONS

The present paper investigated into the problem of modification of the vibrato extent assuming that segments with vibrato are marked by a preceding analysis. The proposed signal operators are entirely based on spectral envelope smoothing operations, and therefore they are conceptually relatively simple. Especially they do not require a manipulation of individual partial parameters. Despite this simplicity the operators did allow us to achieve perceptually very convincing results when applied to vibrato manipulation. They have been evaluated on a task dealing with the removal of vibrato from a flute signal. The results demonstrate that for the flute signal the most complex operator is required to achieve optimal and natural results because the harmonic and noise envelope components do not follow the same amplitude modulation pattern. The requirement for the additional complexity that is related to the separate treatment of noise and sinusoidal modulations depends on the signal - it can be avoided if the noise component is weak. Further studies are needed that deal with the connection of the mod-

ified vibrato to the neighboring note transitions. In the near future we will investigate into generative instrument specific models that relate vibrato and timbre modulation and we will investigate into signal transformation operators that allow manipulating the vibrato rate.

6. ACKNOWLEDGMENTS

Many thanks to composer Alain Lithaud who provided the flute recordings for this investigation.

7. REFERENCES

- [1] M. Schröder, "Emotional speech synthesis: A review," in *Proc. EuroSpeech*, 2001, pp. 561–564.
- [2] F. Burkhardt, "Emofilt: the simulation of emotional speech by prosody-transformation," in *Interspeech*, 2005, pp. 509–512.
- [3] J. Tao, Y. Kang, and A. Li, "Prosody conversion from neutral speech to emotional speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1145–1154, 2006.
- [4] M. Bulut, S. Lee, and S. Narayanan, "Analysis of emotional speech prosody in terms of part of speech tags," in *Interspeech*, 2007, pp. 626–629.
- [5] C.-H. Wu, C.-C. Hsia, C.-H. Lee, and M.-C. Lin, "Hierarchical prosody conversion using regression-based clustering for emotional speech synthesis," *IEEE Trans. Audio, Speech and Lang. Proc.*, vol. 18, no. 6, pp. 1394–1405, 2010.
- [6] E. Lindemann, "Music synthesis with reconstructive phrase modeling," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 80–91, 2007.
- [7] G. Beller, *Analyse et Modèle Génératif de l'Expressivité, Application à la Parole et à l'Interprétation Musicale*, Ph.D. thesis, Université Paris VI UPMC, 2009.
- [8] G. Beller, "Expresso: Transformation of expressivity in speech," in *Proc. Speech Prosody*, 2010.
- [9] R. Maher and J. Beauchamp, "An investigation of vocal vibrato for synthesis," *Applied Acoustics*, vol. 30, pp. 219–245, 1990.
- [10] E. Tellman, L. Haken, and B. Holloway, "Timbre morphing of sounds with unequal numbers of features," *Journal of the Audio Engineering Society*, vol. 43, no. 9, pp. 678–689, 1995.

- [11] D. Arfib and N. Delprat, "Selective transformations of sounds using time-frequency representations - application to the vibrato modification," AES Preprint 4652, 1998.
- [12] M. Mellody and G.H. Wakefield, "The time-frequency characteristics of violin vibrato: Modal distribution analysis and synthesis," *Jour. of the Acoustic Society of America*, vol. 107, no. 1, pp. 598–611, 2000.
- [13] S. Marchand and M. Raspaud, "Enhanced time-stretching using order-2 sinusoidal modeling," in *Proc. 7th Int. Conf. on Digital Audio Effects (DAFx)*, 2004, pp. 76–82.
- [14] V. Verfaillie, C. Guastavino, and P. Depalle, "Perceptual evaluation of vibrato models," in *Proceedings of the conference on interdisciplinaire musicology (CIM05)*, 2005.
- [15] V. Verfaillie, C. Guastavino, and P. Depalle, "Évaluation de modèles de vibrato," *Cahiers de la Société Québécoise de Recherche en Musique*, vol. 9, no. 1/2, pp. 49–62, 2007.
- [16] V. Verfaillie, C. Guastavino, and P. Depalle, "Control parameters of a generalized vibrato model with modulations of harmonics and residual," in *Acoustics'08*, 2008.
- [17] S. Rossignol, X. Rodet, P. Depalle, J. Soumagne, and J.-L. Collette, "Vibrato: detection, estimation, extraction, modification," in *Proc. 2nd Int. Conf. on Digital Audio Effects (DAFx)*, 1999.
- [18] P. Herrera and J. Bonada, "Vibrato extraction and parameterization in the spectral modeling synthesis framework," in *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx)*, 1998.
- [19] R. Timmers and P. Desain, "Vibrato: the questions and answers from musicians and science," in *Proceedings of the Sixth International Conference on Music Perception and Cognition*, 2000.
- [20] J.I. Shonle and K.E. Horan, "The pitch of vibrato tones," *Journal Acoust. Soc. Am.*, vol. 67, no. 1, pp. 246–252, 1980.
- [21] C. d'Alessandro and M. Castellengo, "The pitch of short-duration vibrato tones," *Journal Acoustic Society America*, vol. 95, no. 3, pp. 1617–1631, 1994.
- [22] A. Röbel, F. Villavicencio, and X. Rodet, "On cepstral and all-pole based spectral envelope modeling with unknown model order," *Pattern Recognition Letters, Special issue on Advances in Pattern Recognition for Speech and Audio Processing*, pp. 1343–1350, 2007.
- [23] A. Röbel and X. Rodet, "Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation," in *Proc. of the 8th Int. Conf. on Digital Audio Effects (DAFx05)*, 2005, pp. 30–35.
- [24] X. Amatriain, J. Bonada, A. Loscos, and X. Serra, "Spectral processing," in *Digital Audio Effects*, U. Zölzer, Ed., chapter 10, pp. 373–438. John Wiley & Sons, 2002.
- [25] R. J. McAulay and T. F. Quatieri, "Speech analysis-synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.
- [26] A. Röbel, "Frequency slope estimation and its application for non-stationary sinusoidal parameter estimation," in *Proc. of the 10th Int. Conf. on Digital Audio Effects (DAFx'07)*, 2007.

HARPSICHORD SOUND SYNTHESIS USING A PHYSICAL PLECTRUM MODEL INTERFACED WITH THE DIGITAL WAVEGUIDE

Chao-Yu Jack Perng,

Dept. of Physics, and
Center for Computer Research
in Music and Acoustics,
Stanford University,
Stanford, CA, U.S.A.
perng@stanford.edu

Julius Smith,

Center for Computer Research
in Music and Acoustics,
Stanford University,
Stanford, CA, U.S.A.
jos@ccrma.stanford.edu

Thomas Rossing,

Center for Computer Research
in Music and Acoustics,
Stanford University,
Stanford, CA, U.S.A.
rossing@ccrma.stanford.edu

ABSTRACT

In this paper, we present a revised model of the plectrum-string interaction and its interface with the digital waveguide for simulation of the harpsichord sound. We will first revisit the plectrum body model that we have proposed previously in [1] and then extend the model to incorporate the geometry of the plectrum tip. This permits us to model the dynamics of the string slipping off the plectrum more comprehensively, which provides more physically accurate excitation signals. Simulation results are presented and discussed.

1. INTRODUCTION

The harpsichord is a plucked string keyboard instrument which was first invented probably around the late 14th century [2]. A predecessor of the piano, its popularity reached its peak in the 17th century, becoming one of the most important keyboard instruments of the Baroque era. The harpsichord became "obsolete" rather quickly after the maturation of the piano, but the 20th century early music movement has since renewed significant interest towards the instrument. Figure 1 shows the mechanism in which the harpsichord strings are sounded. When the key is played, the harpsichord jack is guided to move vertically upwards and a flexible plectrum mounted at the end of the jack plucks the string.

General harpsichord physics have been discussed in [3, 4, 5] discussing the various components of the harpsichord. More specific studies such as the soundboard vibration modes or attack transients can be found in [6, 7, 8, 9, 10, 11]. The dynamics of the harpsichord, generally thought to be nonexistent, has been studied in greater detail in [12] and has shown actually that a limited amount of dynamics and timbral changes exist.

The interaction between the harpsichord plectrum and string is an aspect much less studied. A theoretical model was first proposed by Griffel [13], prompting further studies and a modified

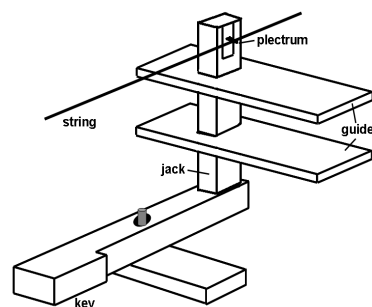


Figure 1: Harpsichord key and jack.

plectrum model proposed by Giordano and Winans II [14]. In contrast, the finger-string interaction has been studied and modeled in more detail in both the guitar [15, 16, 17, 18, 19] and concert harp [20, 21]. For the guitar, differences in radiated sound due to changes in guitar plectrum parameters have been reported in [22, 23], and a guitar plectrum-string interaction model can be found in [24].

A more thorough harpsichord plectrum has been recently proposed by the authors [1, 25], which excites both transverse motions of the strings and allows for interfacing with digital waveguides [26, 27]. In this paper, we extend our model to incorporate the plectrum tip geometry, describing the final stages of the string slipping off the plectrum more completely, important in synthesizing a more accurate string excitation signal. The physical-based excitations provide controllability and expressivity that can complement existing models using a sampled excitation database [28].

In Section 2 we present our improved plectrum model, which includes a review of our previous plectrum body model and the new plectrum tip model. In Section 3 we discuss the plectrum-string interaction, where the interaction at the plectrum body and

tip are treated differently. In Section 4, we detail the interfacing between our plectrum model with the digital waveguide. Simulation results are discussed in Section 5, and in Section 6 we draw our conclusions.

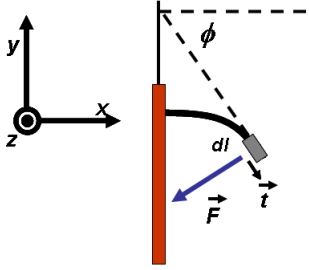


Figure 2: Force exerted on plectrum.

2. PLECTRUM MODEL

In this section, we will first review the plectrum body model. We will then extend our plectrum model so that it incorporates the tip geometry of the plectrum, important for an accurate description of the string during slip-off from the plectrum.

2.1. Model Assumptions

For our model, we shall assume that

- the strain is small within the plectrum (still allows for large end deflection)
- the plectrum is an isotropic elastic material
- the plectrum has a uniform rectangular cross section
- the plectrum only bends in a plane so that there is no twisting motion
- there is no friction between the string and plectrum
- the force exerted on the plectrum is always perpendicular to the surface in contact
- the force is concentrated only at one point
- the plectrum mass is ignored, and thus the plectrum and string are assumed to be quasi-static, neglecting any oscillations from the plectrum's inertia.

2.2. Plectrum Body

The harpsichord plectrum is modeled as a thin rectangular rod clamped at one end and free on the other end. When a clamped rod is subject to an external force \vec{F} , it results in a bending moment \vec{M} due to the internal stresses. The general equilibrium equation for a bent rod is given by

$$\frac{d\vec{M}}{dl} = \vec{F} \times \vec{t} \quad (1)$$

where dl is an infinitesimal element of the rod, and \vec{t} is a unit vector tangential to the rod. Under the assumptions in the previous section, the bending moment can be written as

$$M = EI \frac{d\vec{r}}{dl} \times \frac{d^2\vec{r}}{dl^2} \quad (2)$$

where E is the Young's Modulus, I is the second moment of inertia (or area moment of inertia), and \vec{r} is the radius vector from a fixed point to the point considered on the rod. Defining a coordinate axis such that the x - y plane denotes the plane of the bent rod and ϕ as the angle between the horizontal and \vec{t} , shown in Figure 2, the equation is simplified to

$$EI \frac{d^2\phi}{dl^2} + F = 0 \quad (3)$$

Imposing the correct boundary conditions at the free end, we can solve for the deflection angle along the length of the plectrum,

$$\phi(l) = \frac{F}{EI} \left(Ll - \frac{1}{2} l^2 \right) \quad (4)$$

$$\phi_0 \equiv \phi(l = L) = \frac{1}{2} \frac{FL^2}{EI} \quad (5)$$

where L is length of the plectrum. The parametric shape of the plectrum $x(l)$ and $y(l)$ can be found as

$$\begin{cases} x(l) = \int \cos\phi \, dl \\ y(l) = - \int \sin\phi \, dl \end{cases} \quad (6)$$

For small-angle approximations ($\phi \ll 1$), this reduces to the commonly seen cantilever beam loading equations. A bent harpsichord plectrum, however, undergoes significant deflection, and these cantilever beam equations do not agree well with the general solution (6). A revised approximation that the authors have proposed is given by

$$\begin{cases} x(l) = l - \frac{1}{2} \left(\frac{F}{EI} \right)^2 \left(\frac{L^2 l^3}{3} - \frac{Ll^4}{4} + \frac{l^5}{20} \right) \\ y(l) = - \left(\frac{F}{EI} \right) \left(\frac{Ll^2}{2} - \frac{l^3}{6} \right) \\ \quad + \frac{1}{6} \left(\frac{F}{EI} \right)^3 \left(\frac{L^3 l^4}{4} - \frac{3Ll^5}{10} + \frac{Ll^6}{8} - \frac{l^7}{56} \right) \end{cases} \quad (7)$$

which gives good agreement even up to end deflection angles ϕ of 45° .

2.3. Plectrum Tip

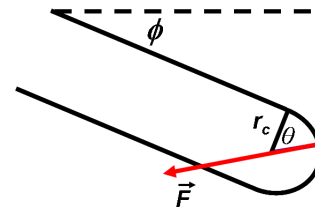


Figure 3: Force exerted on plectrum tip.

In order to account for the geometry of the end of the plectrum, we will model the plectrum tip as a circular tip with diameter equal

to that of the thickness of the plectrum. We will go through derivations similar to that of the previous section 2.2. Also keeping in mind that an exerted force on the tip of the plectrum must still be perpendicular to the surface, as in Fig 3, equation (3) becomes

$$EI \frac{d^2 \phi}{dl^2} + F(\cos \theta) = 0 \quad (8)$$

where θ denotes the angle and position on the tip the force is applied. The deflection angles become

$$\phi(l) = \frac{F(\cos \theta)}{EI} (Ll - \frac{1}{2}l^2) \quad (9)$$

$$\phi_0 \equiv \phi(l = L) = \frac{1}{2} \frac{F(\cos \theta)L^2}{EI} \quad (10)$$

Similarly, for the revised approximation of the plectrum shape, all the F terms are replaced with $F(\cos \theta)$. Note that when $\theta = 90^\circ$, there is no bending moment on the plectrum, and the plectrum becomes unbent with zero deflection $\phi(l) = 0$. In the case of the harpsichord plectrum and string, this represents the moment when the string slides past and leaves the plectrum. It is also clear from the figure that θ will not be larger than 90° , as this would imply that the plectrum is bent upwards instead.

3. PLECTRUM-STRING INTERACTION

In this section, we will discuss the interaction between the harpsichord plectrum and string while they are in contact when the string is plucked. Assuming small string displacements, a segment of the string with mass Δm and length Δz that is in contact with the plectrum follows the equations of motion,

$$\begin{cases} (\Delta m) \frac{\partial^2 x_s(t)}{\partial t^2} = K \frac{\partial^2 x_s(t)}{\partial z^2} (\Delta z) + F_{p-x} \\ (\Delta m) \frac{\partial^2 y_s(t)}{\partial t^2} = K \frac{\partial^2 y_s(t)}{\partial z^2} (\Delta z) + F_{p-y} \end{cases} \quad (11)$$

where $x_s(t)$ and $y_s(t)$ denote the transverse string segment displacements, K is the tension of the string, F_{p-x} and F_{p-y} are the x and y components of the plectrum force F exerted on the string segment, and z is the coordinate along the string, perpendicular to both $x_s(t)$ and $y_s(t)$. The time when the string is sliding along the main plectrum body and when it is slipping off the tip must be treated differently.

3.1. Sliding Along Plectrum Body

As shown in Figure 4, the clamped end of the plectrum moves with the harpsichord jack, constrained to move only in the vertical direction. Its position is denoted by $(x_j(t), y_j(t))$. During the phase where the string is sliding along the plectrum body, the string is at a distance $L' < L$ from the clamped end. Using our revised approximation of equation (7) evaluated at the location of the string $l = L'$, we have

$$\begin{cases} x_s(t) - x_j(t) = L' - \left(\frac{F}{EI} \right)^2 \frac{L'^5}{15} \\ y_s(t) - y_j(t) = - \left(\frac{F}{EI} \right) \frac{L'^3}{3} + \left(\frac{F}{EI} \right)^3 \frac{L'^7}{105} \end{cases} \quad (12)$$

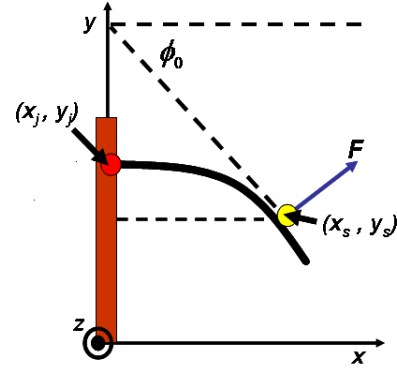


Figure 4: Plectrum and string interaction along main plectrum body.

The deflection angle at $l = L'$ is given by equation (5), and therefore the components of the plectrum force are given by

$$\begin{cases} F_{p-x} = F \sin \left(\frac{FL'^2}{2EI} \right) \\ F_{p-y} = F \cos \left(\frac{FL'^2}{2EI} \right) \end{cases} \quad (13)$$

If we know the motion of the harpsichord jack, the transverse string segment displacements can be calculated using equations (11) to (13).

3.2. Slip-off

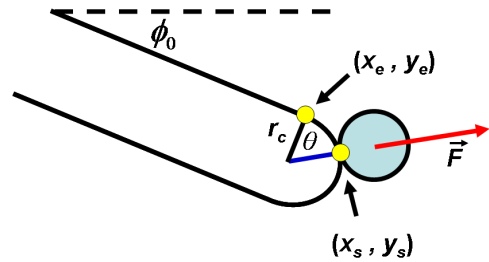


Figure 5: Plectrum and string interaction at plectrum tip.

As shown in Figure 5, as the string slides past the end of the plectrum body, labeled in the figure as point (x_e, y_e) , it proceeds to slip off the tip. While the string is on the plectrum tip, it is a distance $L'' = L + \Delta$ from the clamped end of the plectrum, where Δ is the additional length correction from the tip. However, Δ is on the order of the thickness of the plectrum, which is much smaller than the length of the plectrum. To reduce the complexity of the problem, we will first make the approximation that the force F is applied at the point (x_e, y_e) . Using the plectrum tip model of section 2.3, the deflection of the plectrum at (x_e, y_e) is given

by

$$\begin{cases} x_e(t) - x_j(t) = L - \left(\frac{F(\cos \theta)}{EI} \right)^2 \frac{L^5}{15} \\ y_e(t) - y_j(t) = - \left(\frac{F(\cos \theta)}{EI} \right) \frac{L^3}{3} + \left(\frac{F(\cos \theta)}{EI} \right)^3 \frac{L^7}{105} \end{cases} \quad (14)$$

From the geometry in Figure 5, we also find that

$$\begin{cases} x_s(t) - x_e(t) = r_c [\sin(\theta + \phi_0) - \sin(\phi_0)] \\ y_s(t) - y_e(t) = r_c [\cos(\theta + \phi_0) - \cos(\phi_0)] \end{cases} \quad (15)$$

where r_c is the radius of curvature of the tip, equal to half the plectrum thickness. Combining these two expressions with equation (10), we have the plectrum deflection at the location of the string which accounts for the additional length correction of the plectrum tip:

$$\begin{cases} x_s(t) - x_j(t) = L \left(1 - \frac{4\phi_0^2}{15} \right) + r_c [\sin(\theta + \phi_0) - \sin(\phi_0)] \\ y_s(t) - y_j(t) = - \frac{2L\phi_0}{3} \left(1 - \frac{4\phi_0^2}{35} \right) \\ \quad + r_c [\cos(\theta + \phi_0) - \cos(\phi_0)] \end{cases} \quad (16)$$

Note that if $\theta = 0$, this expression reduces to equation (14), which represents the string just at the edge of the plectrum body. Similarly, the components of the plectrum force are now

$$\begin{cases} F_{p_x} = F \sin(\theta + \phi_0) \\ F_{p_y} = F \cos(\theta + \phi_0) \end{cases} \quad (17)$$

and therefore the the transverse string segment displacements can be calculated once again.

4. DIGITAL WAVEGUIDE INTERFACE WITH PLECTRUM MODEL

For segments of the string not in contact with the plectrum, the equations of motion (11) are reduced to the wave equation

$$\begin{cases} \frac{\partial^2 x_s(t)}{\partial t^2} = c^2 \frac{\partial^2 x_s(t)}{\partial z^2} \\ \frac{\partial^2 y_s(t)}{\partial t^2} = c^2 \frac{\partial^2 y_s(t)}{\partial z^2} \end{cases} \quad (18)$$

where $c = \sqrt{K/\mu}$ is the string wave propagation speed, K is the string tension defined earlier, and $\mu = (\Delta m)/(\Delta z)$ is the linear mass density. D'Alembert's traveling-wave solution to the wave equation is well-known and can be expressed as

$$\begin{cases} x_s(z, t) = x^-(z + ct) + x^+(z - ct) \\ y_s(z, t) = y^-(z + ct) + y^+(z - ct) \end{cases} \quad (19)$$

where x^- and y^- represent the traveling waves in the $-z$ direction and x^+ and y^+ in the $+z$ direction. In the discrete-time domain, traveling waves are simulated efficiently by means of digital

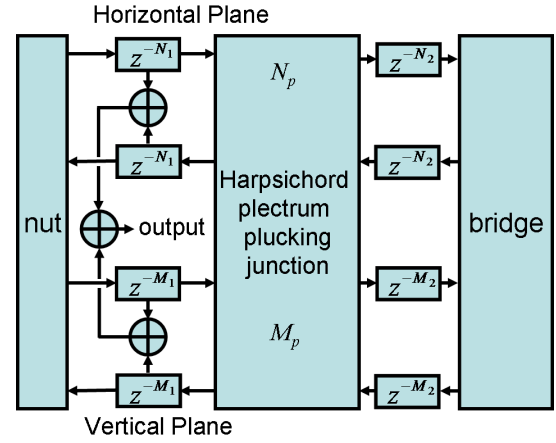


Figure 6: Harpsichord string synthesis model.

waveguides. The harpsichord synthesis model is represented by the block diagram in Figure 6. Pairs of digital waveguide delay-lines are implemented on both sides of the plucking junction. In addition, our plectrum model excites both the horizontal and vertical transverse modes of string vibrations, and the two modes can be coupled both at the nut and the bridge.

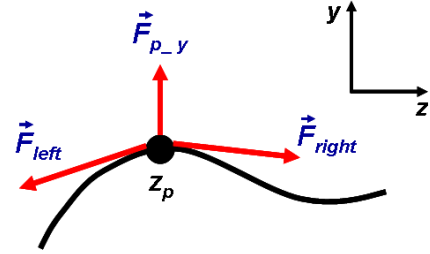


Figure 7: Sum of forces at the plucking point.

The length of the segment of string in contact with the plectrum Δz is much smaller than the length of the string L_s and can be effectively reduced to a single point. As shown in Figure 7 for the transverse vertical y component, the equilibrium of the sum of the forces on the plucking point gives

$$\vec{F}_{p_y} + \vec{F}_{left} + \vec{F}_{right} = 0 \quad (20)$$

For small displacements, the y component of the left and right string forces can be approximated as

$$\begin{cases} F_{left_y} \approx -K \left. \frac{\partial y_s}{\partial z} \right|_{z_p^-} \\ F_{right_y} \approx K \left. \frac{\partial y_s}{\partial z} \right|_{z_p^+} \end{cases} \quad (21)$$

As with the traveling wave solution (19) of string displacements, the left and right string forces can also be decomposed as left and

right traveling “force waves.” Defining the force wave as

$$f^\mp(z \pm ct) \equiv -K \frac{\partial y^\mp(z \pm ct)}{\partial z} \quad (22)$$

the traveling wave decomposition of the forces gives

$$\begin{cases} F_{left_y} &= f_{left}^-(z+ct) + f_{left}^+(z-ct) \\ F_{right_y} &= -f_{right}^-(z+ct) - f_{right}^+(z-ct) \end{cases} \quad (23)$$

Further relating the spatial and time partial derivatives of the force waves, we have expressions for the “Ohm’s Law” for traveling waves

$$\begin{cases} f^- &= -K \frac{\partial y^-}{\partial z} = -\frac{K}{c} \frac{\partial y^-}{\partial t} = -Rv^- \\ f^+ &= -K \frac{\partial y^+}{\partial z} = \frac{K}{c} \frac{\partial y^+}{\partial t} = Rv^+ \end{cases} \quad (24)$$

where $R = K/c = \sqrt{K\mu}$ is the wave impedance of the string, and v^- and v^+ are the traveling velocity wave components of the transverse string velocity. Rewriting equation (20) in terms of the traveling force waves and also using the Ohm’s Law for traveling waves,

$$(f_{left}^- + f_{left}^+) - (f_{right}^- + f_{right}^+) + F_{p_y} = 0 \quad (25)$$

$$R(-v_{left}^- + v_{left}^+) - R(-v_{right}^- + v_{right}^+) + F_{p_y} = 0 \quad (26)$$

In addition, at the plucking point, the left and right transverse string velocities must be continuous,

$$v_{left}^- + v_{right}^+ = v_{right}^- + v_{left}^+ \equiv v \quad (27)$$

where v is defined as the transverse velocity of the plucked point. Equations (26) and (27) allow us to solve for the outgoing velocity waves v_{left}^- and v_{right}^+ in terms of the incoming velocity waves v_{right}^- and v_{left}^+ :

$$\begin{cases} v_{left}^- = v_{right}^- + \frac{F_{p_y}}{2R} \\ v_{right}^+ = v_{left}^+ + \frac{F_{p_y}}{2R} \end{cases} \quad (28)$$

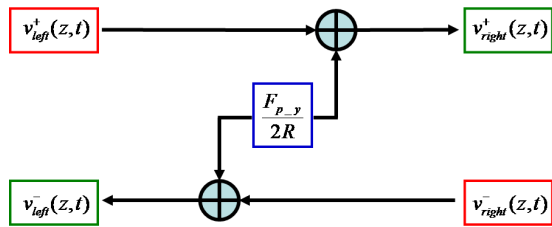


Figure 8: Plectrum plucking junction.

This plucking junction is shown in the diagram of Figure 8. The transverse displacement waves can be evaluated using a Backward Euler method:

$$\begin{cases} y_{left}^-(n) = y_{left}^-(n-1) + v_{left}^-(n)T \\ y_{right}^+(n) = y_{right}^+(n-1) + v_{right}^+(n)T \end{cases} \quad (29)$$

where T is the sampling interval. The transverse x displacement follows an identical derivation. When the string slides off the plectrum, $F_{p_x} = F_{p_y} = 0$, the plucking junction disappears, and the digital waveguide segments to the left and right of the junction are effectively combined into one.

5. RESULTS

5.1. Simulation Parameters

Table 1: Delrin harpsichord plectrum and steel string values.

Plectrum Parameters	
Length L	6 mm
Width W	4 mm
Thickness H	0.5 mm
Second moment of inertia I	0.029 mm ⁴
Young’s modulus E	5 GPa
String Parameters	
Tension T	135 N
Density ρ	7850 kg/m ³
Diameter d	0.37 mm
Linear density μ	0.84 g/m
Length L_s	0.5 m

Modern harpsichord plectra are made out of a plastic material called Delrin. The plectrum and steel string parameters are listed in Table 1. The sampling frequency was chosen at $f_s = 100$ kHz. The plectrum width was made to equal that of one spatial sampling interval $X = 4.0$ mm. The harpsichord jack was assumed to move at a constant velocity v_j :

$$y_j(t) = v_j t$$

Referring to Figure 6 of the synthesis model, while the nut was treated as a rigid termination, we implemented a bridge filter that consisted of a one-pole filter and ripple filter similar to the one implemented in [28]. The transverse x and y string vibrations were not coupled together. That is a direction for future work.

5.2. String Excitation Motion

Figure 9 shows the transverse x and y string motion before the release of the string off the plectrum, plucked at the midpoint with a jack velocity $v_j = 0.02$ m/s. Clearly noticeable is a sharp steep rise in the horizontal displacement just prior to the release of the string that is absent in the vertical string displacement. This corresponds to the slip-off phase when the string is sliding off the plectrum tip. An expanded view of the slip-off portion is shown in Figure 10. This “kick” in the horizontal direction contributes to the brightness of the synthesized harpsichord tone.

5.3. Plucking Speed

Conventional wisdom has it that regardless of how fast one presses on the harpsichord key, the dynamics do not change considerably. Figure 11 shows a graph of the string release amplitude (defined as $A = \sqrt{x_s^2 + y_s^2}$ immediately before the release of the string from the plectrum) v.s. the jack velocity, plucked at the midpoint of the string. Under regular playing speeds of 0.02 – 0.1 m/s, the

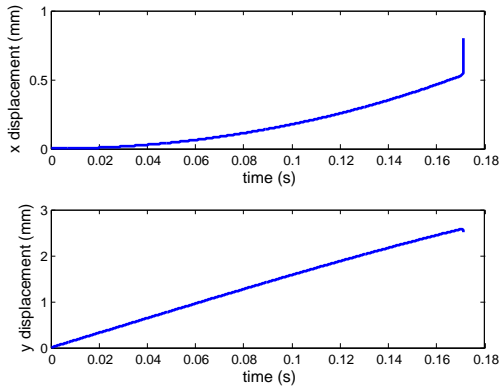


Figure 9: Motion of string before release from plectrum, plucked at the midpoint with jack velocity $v_j = 0.02$ m/s.

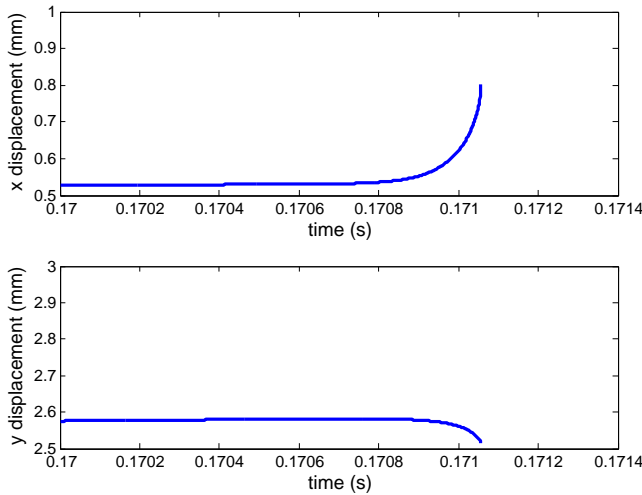


Figure 10: Expanded view of the slip-off of of Figure 9.

amplitude does not vary more than 10%, a difference not readily audible from our simulations. At higher playing speeds, there is a significant increase and drop in the release amplitude, but these are unphysical in the realm of harpsichord playing. In Figure 11 this "peak velocity" occurs at around 2 m/s. For longer bass strings, simulations show a lower the peak velocity but it remains well above reported playing speeds.

5.4. Plucking Point

Many harpsichords have more than one set of strings (called registers) for the same note, where the jacks pluck at different locations along the string. While Italian harpsichords generally had their plucking locations closer together for uniformity of sound, harpsichords built north of the Alps had their strings plucked at locations further apart to create differences in timbre [2]. It is well-established that plucking closer to the nut excites more harmonics

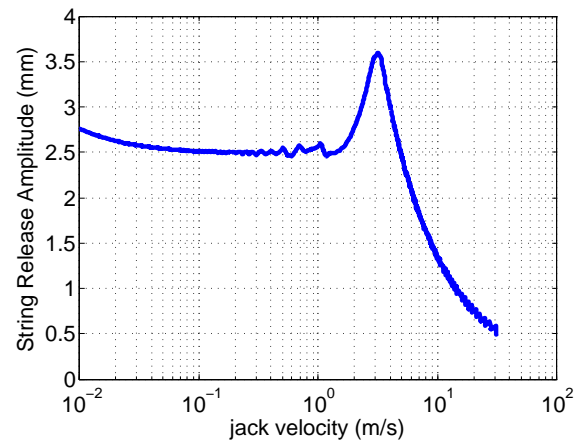


Figure 11: String release amplitude v.s. jack velocity, plucked at the midpoint.

and contributes to a nasal quality to the sound. Our simulations are consistent with this.

As discussed in [1], playing on harpsichord registers which pluck closer to the nut not only results in changes in timbre but also a decrease in volume. The string is released earlier, and the player experiences a "lighter" touch, as the harpsichord jack does not travel as far before the string is plucked. Figure 12 shows simulation results between the plucking location and string release amplitude. As expected, the largest amplitude occurs when plucked at the midpoint and decreases as the plucking point moves closer toward the nut.

6. CONCLUSION

This paper extends the previous harpsichord plectrum model proposed by the authors to incorporate the plectrum tip geometry. Interfacing with a digital waveguide, the complete plucked string motion, especially the final slip-off, is more accurately described. This is crucial in generating the string excitation signals to create realistic plucked harpsichord tones. Future work can include bridge coupling between the two transverse string vibrations and modeling of the lute stop.

7. ACKNOWLEDGMENTS

We would like to thank Nelson Lee and Peter Lindener for fruitful discussions on topics of this paper. In particular, the experimental results from Nelson's plucked guitar-string data, carried out with the assistance of Antoine Chaigne in Paris, exhibited an impulsive force parallel to the plectrum axis that stimulated discussions which inspired the development of our plectrum tip model.

8. REFERENCES

- [1] C.-Y. J. Perng, J. O. Smith III, T. D. Rossing, "Physical modeling of the harpsichord plectrum-string interaction," in *Proc. of Digital Audio Effects Conf. (DAFx '10)*, Graz, Austria, Sept. 2010, pp. 127-130.

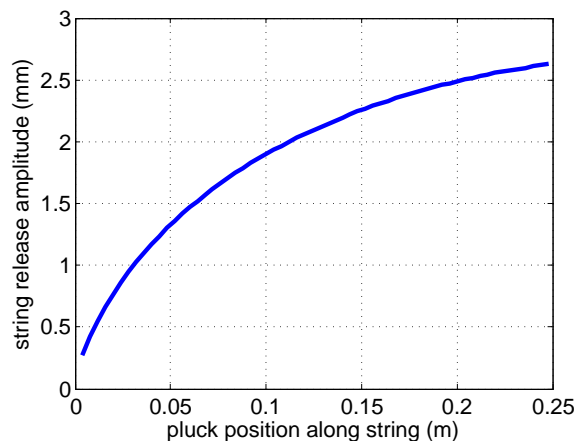


Figure 12: String release amplitude v.s. plucking position, jack velocity $v_j = 0.02$ m/s.

- [2] E. L. Kottick, *A History of the Harpsichord*, Indiana University Press, Bloomington, IN, 2003.
- [3] H. A. Kellner, "Theoretical Physics, The Harpsichord, and its Construction – A Physicists' Annotations," *Das Musikinstrument XXV*, pp.187-194, 1976.
- [4] N. H. Fletcher, "Analysis of the design and performance of harpsichords," *Acustica*, vol. 37, pp.139-147, 1977.
- [5] M. Spencer, "Harpsichord physics," *The Galpin Society Journal*, vol. 34, pp.2-20, 1981.
- [6] E. L. Kottick, "The acoustics of the harpsichord: response curves and modes of vibration," *Galpin Society Journal*, vol. 38, pp. 55-77, 1985.
- [7] E. L. Kottick, K. D. Marshall, T. J. Hendrickson, "The acoustics of the harpsichord," *Scientific American*, vol. 264, pp. 94-99, Feb. 1991.
- [8] W. Savage, E. L. Kottick, T. J. Hendrickson, K. D. Marshall, "Air and structural modes of a harpsichord," *J. Acoust. Soc. Am.*, vol.91, pp. 2180-2189, 1992.
- [9] T. Elfrath, "Determination of the acoustic and vibration-technical characteristics of the harpsichord," Ph.D. thesis, Technical Carolo-Wilhelmina University and Braunschweig, 1992.
- [10] R.-D. Weyer, "Time-frequency-structures in the attack transients of piano and harpsichord sounds-I," & "II" *Acustica*, vol. 35, pp. 232-252, 1976 and vol. 36, pp. 241-258, 1976.
- [11] Hidetoshi Arakawa, "The Acoustical Effect of a Metal Rose in a Harpsichord: Part I," *Proc. of the International Symposium on Musical Acoustics (ISMA2004)*, Nara, Japan, March 31st to April 3rd 2004.
- [12] Henri Penttinen, "On the dynamics of the harpsichord and its synthesis," in *Proc. of Digital Audio Effects Conf. (DAFx '06)*, Montreal, Canada, Sept. 2006, pp. 115-120.
- [13] D. H. Griffel, "The dynamics of plucking," *J. Sound and Vibration*, vol. 175, pp.289-297, 1994.
- [14] N. Giordano and J. P. Winans, II, "Plucked strings and the harpsichord," *J. Sound and Vibration*, vol. 224, pp. 455-473, 1999.
- [15] M. Pavlidou, "A physical model of the string-finger interaction on the classical guitar," Ph.D. thesis University of Wales, U.K., 1997.
- [16] G. Cuzzucoli and V. Lombardo, "A physical model of the classical guitar, including the player's touch," *Computer Music Journal*, vol. 23 no.2, pp. 52-69, Summer 1999.
- [17] F. Eckerholm and G. Evangelista, "The PluckSynth touch string," in *Proc. of Digital Audio Effects Conf. (DAFx '08)*, Helsinki, Finland, Sept. 2008, pp. 213-220.
- [18] G. Evangelista and F. Eckerholm, "Player-instrument interaction models for digital waveguide synthesis of guitar: Touch and collisions," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 18, no.4, pp. 822-832, May 2010.
- [19] G. Evangelista and J. O. Smith III, "Structurally passive scattering element for modeling guitar pluck action," in *Proc. of Digital Audio Effects Conf. (DAFx '10)*, Graz, Austria, Sept. 2010, pp. 10-17.
- [20] J.-L. Le Carrou, F. Gautier, F. Kerjan, J. Gilbert, "The finger-string interaction in the concert harp," in *Proc. of the International Symposium on Musical Acoustics (ISMA2007)*, Barcelona, Spain, September, 2007.
- [21] J.-L. Le Carrou, E. Wahlen, E. Brasseur, and J. Gilbert, "Two dimensional finger-string interaction in the concert harp," in *Proc. Acoustics 08, Paris, France*, 2008, pp. 1495-1500.
- [22] S. Carral and M. Paset, "The influence of plectrum thickness on the radiated sound of the guitar," *Proc. Acoustics '08*, Paris, June 2008.
- [23] S. Carral, "Plucking the string: The excitation mechanism of the guitar," *J. Acoust. Soc. Am.*, vol. 128, no.4, pp. 2448(A), October 2010.
- [24] F. Germain and G. Evangelista, "Synthesis of guitar by digital waveguides: modeling the plectrum in the physical interaction of the player with the instrument," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 18-21, 2009, pp. 25-28.
- [25] C.-Y. J. Perng, T. D. Rossing, M. J. Brown, J. W. Ioup, "Toward the modeling of harpsichord plucking," *J. Acoust. Soc. Am.*, vol. 127, pp. 1733(A), March 2010.
- [26] C.-Y. J. Perng, J. O. Smith III, T. D. Rossing, "Sound synthesis of the harpsichord pluck using a physical plectrum-string interaction model," *J. Acoust. Soc. Am.*, vol. 128, no. 4, pp. 2309(A), October 2010.
- [27] Julius O. Smith III, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74-91, 1992, special issue: Physical Modeling of Musical Instruments, Part I. Available online: <http://ccrma.stanford.edu/~jos/pmudw/>.
- [28] V. Välimäki, H. Penttinen, J. Knif, M. Laurson, and C. Erku, "Sound synthesis of the harpsichord using a computationally efficient physical model," *EURASIP J. Applied Signal Processing*, vol. 2004 no.7, pp. 934-948, 2004.

MODELLING OF BRASS INSTRUMENT VALVES

Stefan Bilbao,

Acoustics and Fluid Dynamics Group/Music

University of Edinburgh

Edinburgh, UK

sbilbao@staffmail.ed.ac.uk

ABSTRACT

Finite difference time domain (FDTD) approaches to physical modeling sound synthesis, though more computationally intensive than other techniques (such as, e.g., digital waveguides), offer a great deal of flexibility in approaching some of the more interesting real-world features of musical instruments. One such case, that of brass instruments, including a set of time-varying valve components, will be approached here using such methods. After a full description of the model, including the resonator, and incorporating viscothermal loss, bell radiation, a simple lip model, and time varying valves, FDTD methods are introduced. Simulations of various characteristic features of valve instruments, including half-valve impedances, note transitions, and characteristic multiphonic timbres are presented, as are illustrative sound examples.

1. INTRODUCTION

The brass family is probably the most studied, among all the musical instruments, from the perspective of pure musical acoustics, and a full list of references would be quite long; for a general overview of the state of the art in brass instrument physics, see [1, 2].

Sound synthesis through physical modeling has developed along various lines; some formulations are based around the now-standard time-domain picture of the brass instrument as a nonlinear excitation coupled to a linear resonator [3], where the tube itself is characterized by its reflection function—among such methods are digital waveguides [4]. If wave propagation is consolidated in delay lines, and if loss and dispersion effects are modelled as lumped, such methods can be extremely efficient. See [5] for recent work on methods related to the waveguide formalism.

Finite difference time domain methods [6], based on direct time/space discretization of the acoustic field, though not as efficient as such structures, allow a very general approach to bore modelling, in particular when more realistic features typical of such instruments are incorporated. One such feature, the action of the brass instrument valve, will be described here; as will be seen, though such effects, necessarily time varying, lead to complications in terms of algorithm design, the resulting computational structure, and associated computational costs, are altered little.

A model of a brass instrument, including a linear model of the bore, a simple excitation mechanism, and valve junctions is presented in Section 2, followed by a description of a simple FDTD scheme in Section 3. Simulation results are presented in Section 4. Synthetic sound examples, created in the Matlab environment, are available at

<http://www2.ph.ed.ac.uk/~sbilbao/brasspage.htm>

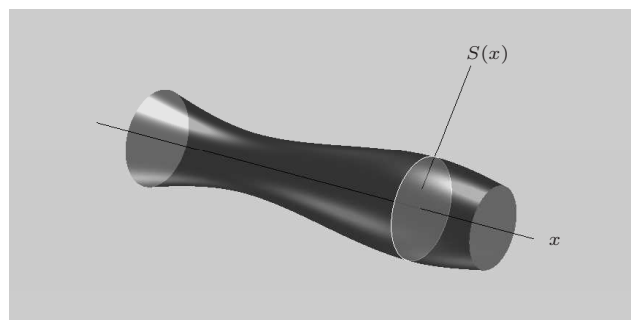


Figure 1: An acoustic tube, of cross-section $S(x)$.

2. BRASS INSTRUMENT MODELS

2.1. Lossless Tubes

Lossless wave propagation in an acoustic tube of variable cross section is described by the following well-known pair of equations:

$$p_t + \frac{\rho c^2}{S} (Sv)_x = 0 \quad (1a)$$

$$v_t + \frac{1}{\rho} p_x = 0 \quad (1b)$$

Here, $p(x, t)$ and $v(x, t)$ are the pressure deviation (from atmospheric) and particle velocity, respectively, at coordinates $x \in [0, L]$ and $t \geq 0$ along a tube of length L m. Subscripts x and t indicate partial differentiation with respect to a spatial coordinate and time, respectively. $S(x)$ is the surface area of the tube at location x , ρ is the density of air, in kg/m^3 and c is the speed of sound, in m/s .

When combined into a single second order equation in p , Webster's equation [7] results. In the interest of keeping the door open to the simulation of distributed nonlinear wave propagation (see [8]), the first order system above will be retained here. Webster's equation, and its variants, are the starting point for physical modeling sound synthesis both in speech [9], including the well-known Kelly-Lochbaum model [10] as well as in brass instruments [11]. The coordinate x will here be taken to be distance along the bore axis—see Figure 1; in a more refined model, it could represent a coordinate normal to isophase surfaces of one-parameter waves, for which there are many choices (see [12]), but the system above remains of essentially the same form.

In this article, since the final structure will be composed of a number of interconnected tubes, system (1) above represents wave

propagation in a single such tube, of length L .

2.2. Lip Model

It is not the purpose of this paper to investigate lip models, which have seen a great deal of theoretical work—see, e.g., [2] for an overview. A standard model is of the following form:

$$\frac{d^2 y}{dt^2} + g \frac{dy}{dt} + \omega^2 (y - H) = \frac{S_r \Delta p}{\mu} \quad (2)$$

In this simple model, the lip displacement $y(t)$, is modelled in terms of a single mass/spring/damper system, driven by a pressure difference Δp across the lips, defined as

$$\Delta p = p_m - p(0, t) \quad (3)$$

where $p_m(t)$ is the blowing pressure, and where $p(0, t)$ is the pressure at the entrance to a tube, the behaviour of which is defined by (1). ω is the angular lip frequency (a control parameter), g is a loss coefficient, and μ is the lip mass (which is sometimes modelled as frequency dependent in the case of lip reed models [13, 14]). S_r is the effective surface area of the lips, and H is an equilibrium opening distance. Further closing relations are

$$u_m = wy \sqrt{\frac{2|\Delta p|}{\rho}} \text{sign}(\Delta p) \quad (4a)$$

$$u_r = S_r \frac{dy}{dt} \quad (4b)$$

$$S(0)v(0, t) = u_m + u_r \quad (4c)$$

where u_m is volume flow at the mouth, u_r is flow induced by the motion of the lips, and where w is the channel width.

The model above is sometimes simplified by assuming $u_r = 0$ [2], or extended through the incorporation of an inertia term [13]; other varieties, including more degrees of freedom for lip motion are also available [14].

2.3. Bell Radiation

A simple model for radiation at the bell, relating particle velocity $v(x = L, t)$ and $p(x = L, t)$ is of the following form:

$$Z_c v = \alpha_1 p + \alpha_2 m \quad p = \frac{dm}{dt} \quad (5)$$

where the parameters α_1 and α_2 , and the characteristic impedance Z_c are defined by

$$\alpha_1 = \frac{1}{4(0.6133^2)} \quad \alpha_2 = \frac{c}{0.6133r} \quad Z_c = \rho c \quad (6)$$

where r is the radius of the bell opening. $m = m(t)$ is an extra lumped variable, reflecting the reactive character of the boundary condition. Such a condition corresponds to a rational (and positive real) approximation to the impedance of an unflanged tube. Such rational approximations are used frequently in speech synthesis [9]; this particular crude approximation matches fairly well with more refined approximations used in brass instrument models [15], and could be improved significantly using a higher order rational approximation [16] (with the important constraint that positive realness is preserved).

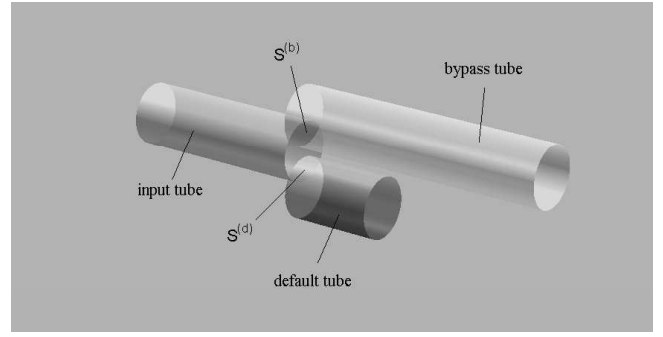


Figure 2: Representation of a junction between an input tube, a default section of tube and a bypass section.

2.4. Valve Junctions

A valve in a brass instrument allows for changes in the effective length of an instrument, through the introduction of an alternate length of tubing. In the most general case of a partly open valve, wave propagation is thus possible both through a short default section of tubing, and a bypass section.

Consider a junction of three tubes, as illustrated in Figure 2. Assuming lossless flow, the pressures at the junction in all three tubes are assumed equal, i.e.,

$$p^{(in)} = p^{(d)} = p^{(b)} \quad (7)$$

where the superscripts (in) , (d) and (b) refer to the input, default and bypass sections, respectively. The volume velocities sum to zero, i.e.,

$$S^{(in)} v^{(in)} = S^{(d)} v^{(d)} + S^{(b)} v^{(b)} \quad (8)$$

where $S^{(in)}$ is the surface area of the input tube at the junction, and where $S^{(d)}$ and $S^{(b)}$ are the overlapping surface areas of the tubes at the junction. For consistency, $v^{(in)}$ is positive when flow is in the direction of the junction, and $v^{(d)}$ and $v^{(b)}$ are negative.

$S^{(d)}$ and $S^{(b)}$ are dependent on the valve state. Because, in either the fully open or closed state, there is not a sizeable discontinuity in the tube cross section, it is useful to parameterize these areas as

$$S^{(d)} = q^{(d)} S^{(in)} \quad S^{(b)} = q^{(b)} S_{(in)} \quad (9)$$

for given values $q^{(d)}(t)$ and $q^{(b)}(t)$, which depend on the current state of the valve at time t . When the tube valve is undepressed, $q^{(d)} = 1$ and $q^{(b)} = 0$, and when depressed, $q^{(d)} = 0$ and $q^{(b)} = 1$.

For a multivalve instrument, then, a single valve is characterised by its location along the bore, the lengths of the default and bypass tubes and its valve state. It can be assumed that the region of the instrument over which the valves are placed is cylindrical, as is the bypass tube, but this is by no means necessary in the FDTD framework.

2.5. Viscothermal Losses

Boundary layer effects in the horn lead to losses which may not be neglected, as they dominate bell radiation losses over the range of playing frequencies for a typical brass instrument—and have a great effect on the impedance curve for the instrument, and thus its playability. They are usually modelled in the frequency domain

through a transmission line formulation (see, e.g., [17, 15], for two slightly different models), and, when converted to the time/space domain, lead to a form similar to (1), but involving new terms:

$$p_t + \frac{\rho c^2}{S} (Sv)_x + fp_{t\frac{1}{2}} = 0 \quad (10a)$$

$$v_t + \frac{1}{\rho} p_x + gv_{t\frac{1}{2}} = 0 \quad (10b)$$

Here, fractional derivative terms have appeared—the accompanying spatially-varying coefficients are

$$f(x) = 2(\alpha - 1)\sqrt{\frac{\eta\pi}{\nu\rho S(x)}} \quad g(x) = 2\sqrt{\frac{\eta\pi}{\rho S(x)}} \quad (11)$$

Here, α is the ratio of specific heats for air, ν is the Prandtl number, and η is the shear viscosity coefficient. See [17] for precise values for these constants. Higher order terms (which play a role only for very thin acoustic tubes) have been neglected here; under further simplification, and after the reduction of the system above to a single second order equation (in p), a form similar to the Webster-Lokshin equation results [18]; the Webster-Lokshin formulation has been used previously in a brass sound synthesis framework [5].

3. FDTD SCHEMES

3.1. Simple Scheme for the Lossless System

System (1) is of the form of a pair of variable transmission line equations (or telegrapher equations [19]), and as such, is analogous to 1D electromagnetic wave propagation, the usual starting point for FDTD methods [20, 21]). An interleaved scheme of the form

$$p_l^n - p_l^{n-1} + \frac{\lambda Z_c}{\bar{S}_l} \left(S_{l+\frac{1}{2}} v_{l+\frac{1}{2}}^{n-\frac{1}{2}} - S_{l-\frac{1}{2}} v_{l-\frac{1}{2}}^{n-\frac{1}{2}} \right) = 0 \quad (12a)$$

$$v_{l+\frac{1}{2}}^{n+\frac{1}{2}} - v_{l+\frac{1}{2}}^{n-\frac{1}{2}} + \frac{\lambda}{Z_c} (p_{l+1}^n - p_{l-1}^n) = 0 \quad (12b)$$

is appropriate. p_l^n is an approximation to $p(x, t)$ at $x = lh$, and $t = nk$, for integer n and l , and for a grid spacing h and time step k ; $f_s = 1/k$ is the sample rate. Similarly, $v_{l+\frac{1}{2}}^{n+\frac{1}{2}}$ is an approximation to $v(x, t)$ at $x = (l + \frac{1}{2})h$, and $t = (n + \frac{1}{2})k$, again for integer n and l . The functions $S_{l+\frac{1}{2}}$ and \bar{S}_l are approximations to $S(x)$ and $x = (l + \frac{1}{2})h$ and $x = lh$, respectively. The characteristic impedance Z_c is as defined in (6), and the numerical parameter λ , or the Courant number [6] is defined as

$$\lambda = ck/h \quad (13)$$

It is possible to show, using either frequency domain or energy techniques [22] that a necessary stability condition for the scheme, if \bar{S}_l is chosen as $\bar{S}_l = (S_{l+\frac{1}{2}} + S_{l-\frac{1}{2}})/2$, is

$$\lambda \leq 1 \quad \rightarrow \quad h \geq ck \quad (14)$$

Generally, it is best to choose h , given k (fixed by the sample rate, which is chosen a priori) as close to this bound as possible, to minimize numerical dispersion, and maximize output bandwidth. See [8] for more on the subject of accuracy. In the special case that $\lambda = 1$, the system above becomes equivalent, upon the introduction of wave variables, to the Kelly-Lochbaum model [10], and, furthermore, when S is constant, to a digital waveguide [4].

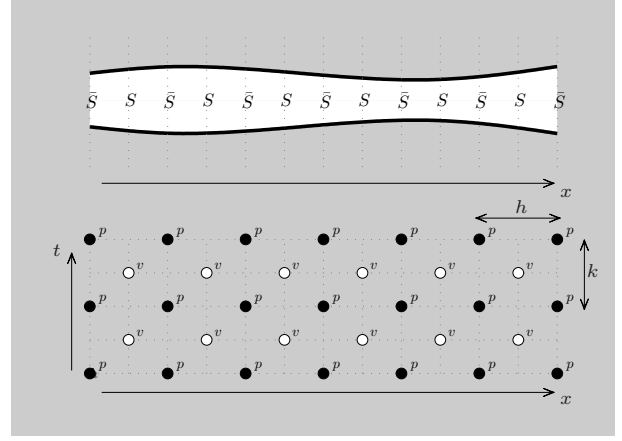


Figure 3: Top: acoustic tube, for which approximations to the surface area S and \bar{S} are made at alternating grid locations. Bottom: interleaved time/space grid for pressure p and velocity v .

For a tube of length L , it is convenient (though by no means necessary) to choose h such that $N = L/h$ is an integer, while satisfying (14). In this case, p_l^n runs over integers $l = 0, \dots, N$, taking on values at the endpoints of the tube, and $v_{l+\frac{1}{2}}^{n-\frac{1}{2}}$ at the interleaved values, from $l = 0, \dots, N-1$. One implication of such a choice, in a network composed of many such tubes (such as a brass instrument) is that one will have a different grid spacing, and associated value of the Courant number λ for each tube segment.

3.2. Termination Conditions

Discretization of the termination conditions is similar to the case of reed wind instruments, discussed in [22], and is reviewed only briefly here.

At the bell termination of the final tube segment, the update is

$$p_N^n - p_N^{n-1} + \frac{\lambda Z_c}{\bar{S}_N} \left(S_{N+\frac{1}{2}} v_{N+\frac{1}{2}}^{n-\frac{1}{2}} - S_{N-\frac{1}{2}} v_{N-\frac{1}{2}}^{n-\frac{1}{2}} \right) = 0 \quad (15)$$

which requires access to the virtual value $v_{N+\frac{1}{2}}^{n-\frac{1}{2}}$. An approximation to the radiation boundary condition (5), namely

$$\begin{aligned} v_{N+\frac{1}{2}}^{n-\frac{1}{2}} + v_{N-\frac{1}{2}}^{n-\frac{1}{2}} &= \frac{\alpha_1}{Z_c} (p_N^n + p_N^{n-1}) + \frac{\alpha_2}{Z_c} (m^n + m^{n-1}) \\ p_N^n + p_N^{n-1} &= \frac{2}{k} (m^n - m^{n-1}) \end{aligned}$$

allows the closure of the system, where an extra update in a time series $m = m^n$ is required. The above boundary condition may be shown to be numerically passive [22], and thus there is no risk of numerical instability.

For the lip model, the various dependent variables may be replaced by time series, i.e., $y^n, \Delta p^n, u_m^{n+\frac{1}{2}}, u_r^{n+\frac{1}{2}}$, approximating the various functions at interleaved multiples of k , the time step. The only time-dependent equations are (2) and (4b), which may be

approximated directly as

$$\frac{1}{k^2} (y^{n+1} - 2y^n + y^{n-1}) + \frac{g}{2k} (y^{n+1} - y^{n-1}) + \frac{\omega^2}{2} (y^{n+1} + y^{n-1}) - H = \frac{S_r \Delta p^n}{\mu} \quad (16)$$

$$u_r^{n+\frac{1}{2}} = \frac{S_r}{k} (y^{n+1} - y^n) \quad (17)$$

3.3. Tube Boundaries

Consider now a junction of three tubes, as described in Section 2.4. In the FDTD setting, with each of the schemes for the three tubes (impinging, default and bypass) is associated a distinct Courant number $\lambda^{(in)}$, $\lambda^{(d)}$ and $\lambda^{(b)}$, all of which should be chosen as close to unity as possible. Furthermore, as pressure values calculated by the three schemes at the junction are coincident, it is straightforward to translate condition (7) to the discrete setting (suppressing time indices), as

$$p_N^{(in)} = p_0^{(d)} = p_0^{(b)} = p^{(J)} \quad (18)$$

where $p^{(J),n}$ is the common pressure at the junction, and where $p_N^{(in),n}$ is the pressure in the impinging tube at its right end, and where $p_0^{(d),n}$ and $p_0^{(b),n}$ are the pressure in the default and bypass sections.

The updates (12a) require access to values of the velocity grid functions at virtual locations not on the respective grids, i.e., $v_{N+\frac{1}{2}}^{(in)}$, $v_{-\frac{1}{2}}^{(d)}$ and $v_{-\frac{1}{2}}^{(b)}$. These can be set according to the flow boundary condition (8) as

$$S_{N-\frac{1}{2}}^{(in)} v_{N-\frac{1}{2}}^{(in)} + S_{N+\frac{1}{2}}^{(in)} v_{N+\frac{1}{2}}^{(in)} = S_{-\frac{1}{2}}^{(in)} v_{-\frac{1}{2}}^{(d)} + S_{\frac{1}{2}}^{(d)} v_{\frac{1}{2}}^{(d)} + S_{-\frac{1}{2}}^{(b)} v_{-\frac{1}{2}}^{(b)} + S_{\frac{1}{2}}^{(b)} v_{\frac{1}{2}}^{(b)} \quad (19)$$

The conditions (18) and (19) may be combined to give a single update for the junction pressure $p^{(J),n}$ as

$$p^{(J),n} - p^{(J),n-1} = \beta^{(J)} \left(S_{N-\frac{1}{2}}^{(in)} v_{N-\frac{1}{2}}^{(in),n-\frac{1}{2}} - S_{\frac{1}{2}}^{(d)} v_{\frac{1}{2}}^{(d),n-\frac{1}{2}} - S_{\frac{1}{2}}^{(b)} v_{\frac{1}{2}}^{(b),n-\frac{1}{2}} \right) \quad (20)$$

where $\beta^{(J)}$ is given by

$$\beta^{(J)} = \frac{2Z_c}{S_N^{(in)} (1/\lambda^{(in)} + q^{(b)}/\lambda^{(b)} + q^{(d)}/\lambda^{(d)})} \quad (21)$$

where $q^{(b)}$ and $q^{(d)}$ follow from the current valve state, as in (9).

The above explicit update for the junction pressure should suggest the analogous update in a scattering framework (such as wave digital filters [23] or digital waveguides [4]). Conversely, it may also be seen that the explicit updating, often claimed to be a benefit of such approaches, is in fact characteristic of direct (i.e., non-scattering) methods as well.

3.4. Filter Designs and Viscothermal Losses

The fractional derivative terms in (10), though standard in frequency domain analysis of acoustic tubes [15] pose some numerical challenges in the FDTD setting, as they do in scattering based approaches. A simple approach (among many; see [24, 5] for other

examples) which has been described recently in [8], is to employ an FIR filter design. To approximate the term $p_{t\frac{1}{2}}$ in (10a), one may employ, generally,

$$p_{t\frac{1}{2}}(x, t) \approx \sum_{m=0}^M a_m p_l^{n-m} \quad (22)$$

for some suitably chosen parameters a_m (perhaps through a frequency domain optimization procedure), and for a chosen order M . In order to get reasonable accuracy at low frequencies, the order M must be chosen to be moderately high—between 20 and 40, at a typical audio sample rate, such as $f_s = 44100$ Hz. A similar approximation may obviously be used for the term $v_{t\frac{1}{2}}$ in (10b). Scheme (12) may thus be generalized to

$$p_l^n - p_l^{n-1} + \frac{\lambda Z_c}{S_l} \left(S_{l+\frac{1}{2}} v_{l+\frac{1}{2}}^{n-\frac{1}{2}} - S_{l-\frac{1}{2}} v_{l-\frac{1}{2}}^{n-\frac{1}{2}} \right) + k f_l \sum_{m=0}^M a_m p_l^{n-m} = 0 \quad (23a)$$

$$v_{l+\frac{1}{2}}^{n+\frac{1}{2}} - v_{l+\frac{1}{2}}^{n-\frac{1}{2}} + \frac{\lambda}{Z_c} (p_{l+1}^n - p_{l-1}^n) + k g_{l+\frac{1}{2}} \sum_{m=0}^M a_m v_{l+\frac{1}{2}}^{n+\frac{1}{2}-m} = 0 \quad (23b)$$

where f_l and $g_{l+\frac{1}{2}}$ are approximations to $f(x)$ and $g(x)$ from (11) at locations $x = lh$ and $x = (l + \frac{1}{2})h$, respectively.

It is important to point out that the order M of the approximation will determine the memory requirement for the algorithm as a whole, and has a strong impact on computational complexity—see Section 3.5. It would thus be advantageous to employ rational filter designs of potentially much lower order.

3.5. Computational Costs

The computational cost of the scheme for the entire system is determined by the total length of tube, L_{total} , which is made up of contributions from the main bore, as well as the bypass tubes. For a Courant number of 1 in all the sections (this is the worst case), the total number of grid points will be $N(f_s) = L_{total} f_s / c$, and thus the total memory requirement, to hold both pressure and velocity variables, will be

$$\text{memory requirement} = 2NM \quad (24)$$

where M is the order of the approximating filter for viscothermal losses. The combined addition+multiplication count per second will be

$$\text{operation count/sec.} = N(4M + 6)f_s \quad (25)$$

At $f_s = 44100$, for a total tube length of $L = 1$ m, and for $M = 20$, the floating point operation rate is on the order of 500 Mflops. This is not cheap, compared with, e.g., a digital waveguide implementation, but neither is it exorbitant, by the standards of today's microprocessors. On the other hand, using such a scheme in a time-varying setting (i.e., employing valve transitions) requires only $O(1)$ additional operations per time step, and is thus of negligible cost.

4. SIMULATION RESULTS

In this section, simulation results are presented for a given bore profile corresponding to a Smith-Watkins trumpet, with a Kelly Screamer mouthpiece. The bore profile is shown in Figure 4. Valves are located at 67.3, 72 and 75 cm along the bore from the mouthpiece end, and the bypass tube lengths are 27 cm, 20 cm and 15 cm, respectively; the default tube lengths are all 2 cm.

4.1. Simulated Impedances

As a preliminary check of the validity of this method, a comparison between a measured input impedance curve for a trumpet, and one computed using an FDTD method, running at 44.1 kHz, is shown in Figure 4. The curves match well over most of the playing range of the instrument, with some deviations apparent above 1 kHz—these are due to the particular simple choice of radiation impedance made here, which underestimates loss at high frequencies, and which could be easily rectified by using a higher order rational approximation.

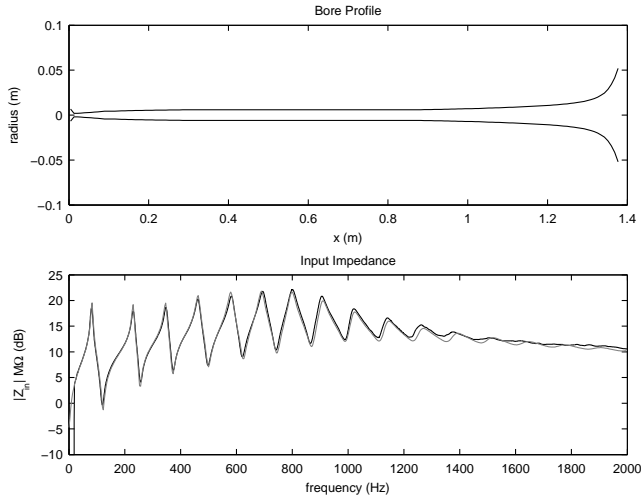


Figure 4: Bore profile, top, for a trumpet, including the mouthpiece, and input impedance magnitudes, bottom, from measurement (black) and simulation (grey).

4.2. Half-valve Impedances

As an example of the behaviour of this numerical method, it is interesting to examine simulated impedance curves, under different half-valve configurations, as illustrated in Figure 5.

4.3. Valve Transitions

In its simplest use, the system described here should be capable of effecting simple changes in pitch. See Figure 6 for a spectrogram of sound output when a single valve is depressed in the model, where the bore profile is that of a trumpet. In this case, the lip parameters and blowing pressure are kept constant, but in a true playing situation, however, one would expect that various parameters (and especially the blowing pressure and lip frequency) are varied simultaneously. Depending on the precise trajectories of

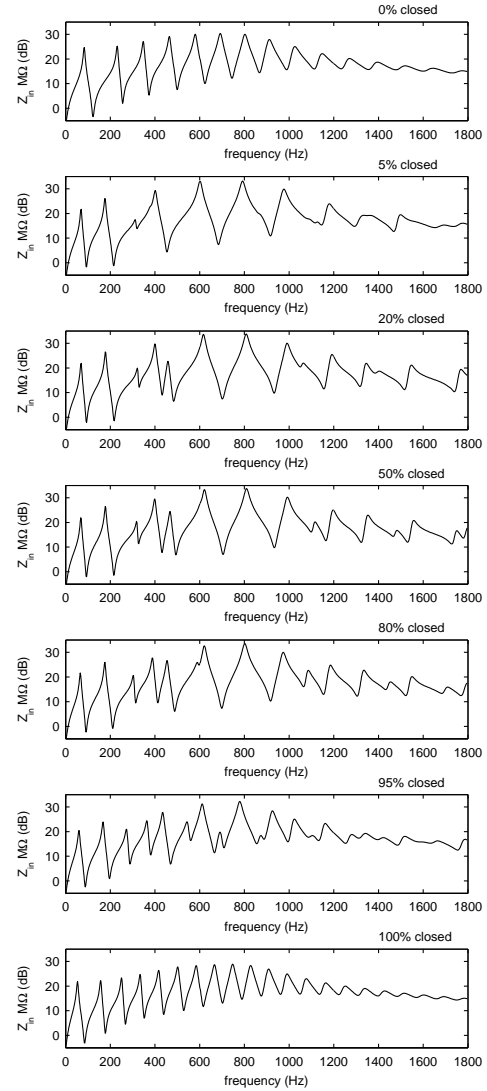


Figure 5: Input impedance magnitudes, for a trumpet, under partially closed valve conditions. Successive plots show impedance magnitudes at varying degrees of simultaneous closure of all three valves (with 100% corresponding to a fully depressed state).

these control signals, one expects a wide variety of possible note transitions, and also situations where the note transition does not occur, and there is rather a noise like timbre, warble, or multi-phonetic results—see Section 4.5.

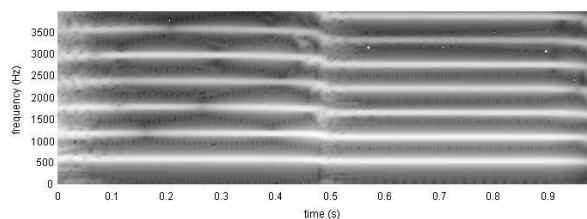


Figure 6: Spectrogram of sound output for a typical valve transition, for a trumpet bore, with a single valve (effecting a change in pitch of a semitone).

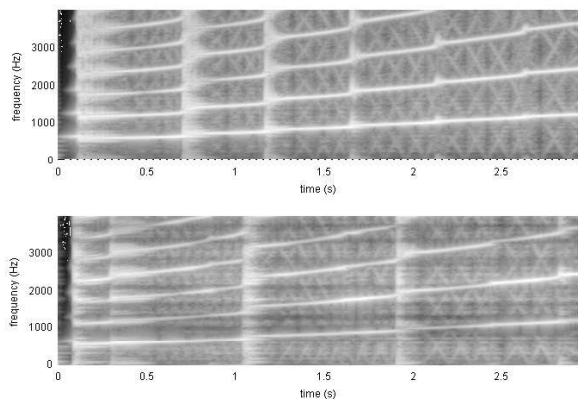


Figure 7: Spectrogram of sound output under a constant linear sweep of the lip frequency, for a trumpet, when all valves are undepressed (top), and half depressed (bottom).

4.4. Glissandi

From Figure 5, illustrating impedance curves under partially closed valve conditions, one may note that when all valves are approximately half open, the resonances over the middle of the playing range become more sparse, and are relatively wide. Under such conditions, a player may more easily effect a glissando than in the case when valves are all in an either fully depressed or undepressed state. See Figure 7, showing spectrograms illustrating a typical gesture under both conditions.

4.5. Warbles and Multiphonics

The irregularity of the impedance curve for an instrument with all valves partially depressed, leads to a wide variety of possible behaviours.

See Figure 8, showing spectrograms of sound output, for a trumpet with all valves half depressed, and for slightly different lip frequencies. At 320 Hz, the instrument produces a pure tone, at 350 Hz, a warble at a sub-audio rate, and at 400 Hz, a noise-like timbre. As can also be observed, note onset times vary considerably with frequency.

4.6. Sound Examples

Sound examples are available on the author's website at

<http://www2.ph.ed.ac.uk/~sbilbao/brasspage.htm>

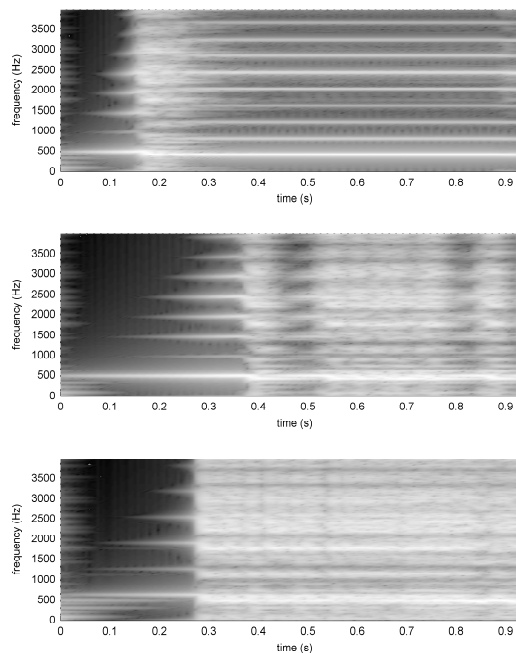


Figure 8: Spectrograms of sound output, for a trumpet, when all valves are half depressed, under different lip frequencies. Top: 320 Hz, middle, 350 Hz and bottom, 400 Hz.

5. CONCLUSIONS AND FUTURE WORK

At the level of the model itself, numerous refinements are possible, which do not alter the basic computational structure described here. Among these are a closer attention to the precise definition of Webster's equation (and an appropriate choice of spatial coordinate), as mentioned in Section 2.1, and an improved model of the radiation impedance, as described in Section 2.3; such refinements lead to relatively minor improvements, in terms of agreement between experiment and simulation (which is already quite good for the model presented here), and may not lead to any discernible benefits in synthesis.

The incorporation of nonlinear effects as described recently in [8], on the other hand, is anticipated to be of major perceptual significance, and requires a more involved treatment (perhaps resorting to finite volume methods [25], and employing artificial viscosity in order to prevent numerical oscillation near the formation of a shock front); even in this case, however, the basic structure of the scheme remains little changed.

As far as the scheme itself is concerned, though the lossless scheme (12) performs very well indeed, the discrete-time approximation to viscothermal losses is rather crude, and leads to a computational bottleneck, both in terms of memory and the over-all operation count. A better approach would perhaps be to use a rational filter approximation; while not problematic in the linear case, such IIR filters, when used to approximate viscothermal losses, will generally exhibit large variations in the coefficient values themselves, and may be difficult to employ in conjunction with a fully

nonlinear model of wave propagation.

Because the scheme is uniform over the entire length of the bore (i.e., grid points are treated equally, and there is not a decomposition into variable length components, or lumping of loss or dispersion effects), programming complexity is quite low for such methods; indeed, in the Matlab code written by the author, the update for the bore in the run time loop may be written in four lines. It is also very well suited to parallelization in a multicore or GPGPU environment.

One aspect of synthesis which has not been discussed here in any detail is that of control. The determination of lip parameters, such as frequency and mass, necessarily time varying, during a playing gesture already presents a difficult experimental challenge; when the extra layer of valve control is also present, the challenges become formidable. Such difficulties are to be expected in any complete physical modeling synthesis framework, and are, in many ways, a measure of the maturity of physical modeling synthesis—beyond building the instrument, one must also learn how to play it. It is thus hoped that the synthesis algorithm presented here will also be useful in scientific studies of brass instrument playing, and such work is under way at the University of Edinburgh.

6. ACKNOWLEDGMENTS

Thanks to Shona Logie, John Chick and Arnold Myers, at the University of Edinburgh, for providing measurements of bore profiles and impedance curves for various brass instruments.

7. REFERENCES

- [1] A. Chaigne and J. Kergomard, *Acoustique des Instruments de Musique*, Belin, Paris, France, 2008.
- [2] N. Fletcher and T. Rossing, *The Physics of Musical Instruments*, Springer-Verlag, New York, New York, 1991.
- [3] M. McIntyre, R. Schumacher, and J. Woodhouse, “On the oscillations of musical instruments,” *Journal of the Acoustical Society of America*, vol. 74, no. 5, pp. 1325–1345, 1983.
- [4] J. O. Smith III, *Physical Audio Signal Processing*, Stanford, CA, 2004, Draft version. Available online at <http://ccrma.stanford.edu/~jos/pasp04/>.
- [5] R. Mignot and T. Hélie, “Acoustic modelling of a convex pipe adapted for digital waveguide simulation,” in *Proceedings of the 13th International Digital Audio Effects Conference*, Graz, Austria, September 2010.
- [6] J. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth and Brooks/Cole Advanced Books and Software, Pacific Grove, California, 1989.
- [7] P. Morse and U. Ingard, *Theoretical Acoustics*, Princeton University Press, Princeton, New Jersey, 1968.
- [8] S. Bilbao, “Time domain modelling of brass instruments,” in *Proceedings of Forum Acusticum*.
- [9] L. Rabiner and R. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.
- [10] J. Kelly and C. Lochbaum, “Speech synthesis,” in *Proceedings of the Fourth International Congress on Acoustics*, Copenhagen, Denmark, 1962, pp. 1–4, Paper G42.
- [11] D. Berners, *Acoustics and Signal Processing Techniques for Physical Modelling of Brass Instruments*, Ph.D. thesis, Department of Electrical Engineering, Stanford University, 1999.
- [12] T. Hélie, “Unidimensional models of acoustic propagation in axisymmetric waveguides,” *Journal of the Acoustical Society of America*, vol. 114, no. 5, pp. 2633–2647, 2003.
- [13] D. Keefe, “Physical modeling of wind instruments,” *Computer Music Journal*, vol. 16, no. 4, pp. 57–73, 1992.
- [14] S. Adachi and M. Sato, “Time-domain simulation of sound production in the brass instrument,” *Journal of the Acoustical Society of America*, vol. 97, no. 6, pp. 3850–3861, 1995.
- [15] R. Caussé, J. Kergomard, and X. Lurton, “Input impedance of brass musical instruments—comparison between experiment and numerical models,” *Journal of the Acoustical Society of America*, vol. 75, no. 1, pp. 241–254, 1984.
- [16] F. Silva, P. Guillemain, J. Kergomard, B. Mallaroni, and A. Norris, “Approximation forms for the acoustic radiation impedance of a cylindrical pipe,” *Journal of Sound and Vibration*, vol. 322, pp. 255–263, 2009.
- [17] D. Keefe, “Acoustical wave propagation in cylindrical ducts: Transmission line parameter approximations for isothermal and nonisothermal boundary conditions,” *Journal of the Acoustical Society of America*, vol. 75, no. 1, pp. 58–62, 1984.
- [18] T. Hélie and D. Matignon, “Diffusive representations for the analysis and simulation of flared acoustic pipes with viscothermal losses,” *Mathematical Models and Methods in Applied Sciences*, vol. 16, no. 4, pp. 503–536, 2006.
- [19] D. Cheng, *Field and Wave Electromagnetics*, Addison-Wesley, Reading, Massachusetts, second edition, 1989.
- [20] A. Taflov, *Computational Electrodynamics*, Artech House, Boston, Massachusetts, 1995.
- [21] K. Yee, “Numerical solution of initial boundary value problems involving Maxwell’s equations in isotropic media,” *IEEE Transactions on Antennas and Propagation*, vol. 14, pp. 302–307, 1966.
- [22] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, John Wiley and Sons, Chichester, UK, 2009.
- [23] A. Fettweis, “Wave digital filters: Theory and practice,” *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.
- [24] J. Abel, T. Smyth, and J. O. Smith III, “A simple, accurate wall loss filter for acoustic tubes,” in *Proceedings of the 6th International Digital Audio Effects Conference*, London, UK, September 2003, pp. 254–258.
- [25] R. Leveque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, Cambridge, UK, 2002.

PHYSICAL MODEL OF THE STRING-FRET INTERACTION

Gianpaolo Evangelista

Digital Media,
Linköping University
Norrköping, Sweden
firstname.lastname@itn.liu.se

ABSTRACT

In this paper, a model for the interaction of the strings with the frets in a guitar or other fretted string instruments is introduced.

In the two-polarization representation of the string oscillations it is observed that the string interacts with the fret in different ways. While the vertical oscillation is governed by perfect or imperfect clamping of the string on the fret, the horizontal oscillation is subject to friction of the string over the surface of the fret.

The proposed model allows, in particular, for the accurate evaluation of the elongation of the string in the two modes, which gives rise to audible dynamic detuning. The realization of this model into a structurally passive system for use in digital waveguide synthesis is detailed.

By changing the friction parameters, the model can be employed in fretless instruments too, where the string directly interacts with the neck surface.

1. INTRODUCTION

Accurate physically inspired synthesis of musical instrument require realistic models of all the parts of the instrument that significantly contribute to the production of the characteristic timber and its evolution, together with sufficiently general models of the interaction of the player with the instrument [1].

This work is a piece of a broader project whose aim is to closely emulate the playing of a guitar, with extension to other instruments in the family of plucked strings. In previous papers, the author, together with other researchers, introduced models for the plucking of the string, both with finger and plectrum, for the collisions of the string with the neck and other objects and for the synthesis of harmonic or flageolet tones [2, 3, 4, 5, 6, 7]. The models were introduced for immediate application in digital waveguide synthesis of the guitar, but they are also usable in other type of synthesis techniques such as finite difference time domain (FDTD).

In this paper, the issue of modeling the fret-string interaction is considered, which influences the sound produced by the synthesis algorithm. Disregarding longitudinal modes, a guitar string is represented by coupled wave equations, each pertaining to a polarization mode, i.e. to one of the orthogonal axes in the planes transversal to the string. In a fretted instrument, when a player's finger pushes the string against the frets in order to produce the desired tone, perfect or near perfect clamping only occurs in the direction normal to the fret surface.

In the horizontal direction, i.e. in the direction parallel to fret and tangent to this – in many electric guitars the fret is curved – the string is free to move but subject to friction on the fret surface. As a result, not only the two polarization modes show different

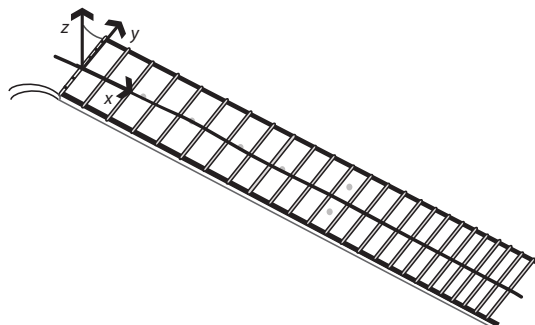


Figure 1: *Coordinate system aligned with the string rest position.*

decay times but also their fundamental frequencies differ and vary with time, due to the unequal elongation of the string. In fretless instruments the dynamic is similar but the string is pushed directly against the neck. As a result, clamping is less perfect in the vertical direction and the string is still free to move but subject to friction on the neck surface in the horizontal direction. Furthermore, the direct contact with the finger of the player introduces a considerable amount of damping and collisions with the neck are more likely due to smaller string to neck distance.

A new model for the fret-string interaction is presented in this paper, which is based on recent advances in modeling friction with dynamic systems [8, 9]. From the equations governing the string motion in the contact area with fret or neck we derive a structurally passive junction to be included as a fret-string interaction module in a pair of double rail digital waveguides, one for each polarization mode.

2. FRETBOARD FINGERING

In this paper we choose the coordinate system shown in Figure 1 where the x axis is directed from nut to bridge along the string rest position. The y axis representing the horizontal direction is parallel to the frets, while the z axis is orthogonal to the other two axes and represents the vertical direction. In order to fix our ideas, we assume that the instrument is designed for a right-hand player, where strings are plucked with the right hand and the player pushes fingers of the left hand on the strings against the fretboard.

In conventional fretted instrument, frets are spaced on the fingerboard in order to achieve equally tempered tuning. This is achieved by placing the active (topmost) edge of the fret at co-

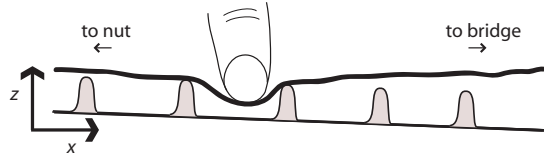


Figure 2: Side view of the vertical shape of the string when pressed against frets by a player's finger.

ordinates

$$x_n = L_s \left(1 - 2^{-n/12}\right), \quad (1)$$

where L_s is the length of the string and n the fret number counted from nut to bridge [10].

Since the bridge is taller than the nut, the neck of the instrument, which supports the fretboard, appears as slightly tilted with respect to the string rest position. As a result, when the player pushes the string against the fretboard, the string rests on two frets (or one fret and the nut for the lowest fingered tone $n = 1$), as shown in Figure 2. The leading fret is the one closest to the bridge (rightmost) and is responsible for tuning by reducing the length of the active portion of the string (from fret to bridge). The trailing fret (leftmost) further blocks residual vibrations from reaching the inactive portion of the string. Occasional collisions with other frets may also occur if the string is vigorously plucked in the vertical direction.

Fingering on the fretboard produces a deflection of the string that slightly modifies the string length. Characteristics influencing the intonation of fretted guitar tones are described in [11]. The player does not need to push the finger all the way against the fretboard: in order to produce proper tones it suffices that the string rests quite firmly on the leading fret. This is generally achieved by placing the finger as close as possible to the leading fret. The frequency of the tone slightly depends on how much the string is pushed towards the fingerboard.

With respect to the vertical polarization mode, the string appears as clamped to the fret. In the horizontal direction, however, the string is quite free to slip over the fret, as shown in Figure 3. The motion is subject to friction force in the direction opposite to velocity and to the restoring tensile forces of the string along the y direction. The horizontal oscillations of the string are further coupled to the finger behind the leading fret, which essentially acts as an elasto-plastic spring damper. Residue oscillations further travel toward the trailing fret, subject to further friction, and toward the nut and back. However, the amplitude of oscillation in this trailing path is negligible due to the damping and clamping introduced by the finger pressing the string towards the neck.

3. FRICTION MODEL

In order to simulate the stick-slip motion of the string over the fret in the horizontal polarization mode, a suitable model for the friction is necessary. A sufficiently general scheme is derived from [8, 12], where the effect of friction is modeled as a dynamic system, known as the Lu-Gre model, which generalizes the Coulomb model of friction. In this model, the surfaces are thought of as being randomly coated by elastic bristles, which deflect as two contacting surfaces are set in relative motion.

An extension [12] of the bristle based model has been previously used in sound synthesis to capture the dynamics of the vio-

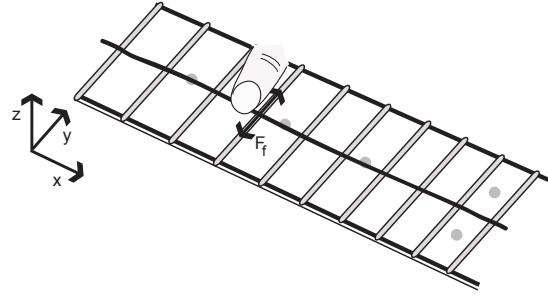


Figure 3: Top view of the guitar fretboard showing string-fret friction force in the horizontal polarization mode of the string.

lin bow [13] or to model general friction interactions among rigid bodies [14] with modal synthesis. Here we apply a similar model to the string-fret interaction and provide a realization for use in digital waveguide simulation of the string.

Although remarkable generalizations of the friction model have been introduced [9], which allow us to capture subtle phenomena such as friction hysteresis, our aim is to obtain a simple system capturing the main characteristics of the string-fret interaction at reasonable computational costs. Unlike in friction driven sound, the friction noise in the fret-string interaction is not a main audible feature but friction does contribute to the dynamics of the string, which is audible through modulation of the elongation and slow-down of the string. Its inclusion contributes to a more naturally sounding model.

3.1. Bristle-Based Friction Models

Following [12], the average deflection ξ of elasto-plastic bristles can be modeled by the following first order differential equation:

$$\frac{d\xi}{dt} = v_{rel} \left(1 - \alpha(v_{rel}, \xi) \frac{\xi}{\xi_{ss}(v_{rel})}\right) \quad (2)$$

where v_{rel} denotes the relative velocity of the contacting surfaces (the string and the fret in our case). The function $\alpha(v, \xi)$ allows us to capture the elasto-plastic behavior of the bristles for large displacement. In a simplified model (Lu-Gre) one can let $\alpha(v_{rel}, \xi) = 1$.

The function $\xi_{ss}(v)$ provides the limit value for the deflection in steady state where the relative velocity v , instantiated by v_{rel} in (2), and the average bristle deflection are constant.

The friction force f_f can be written in terms of bristle displacement and relative velocity as follows:

$$f_f(\xi, \dot{\xi}, v_{rel}) = \sigma_0 \xi + \sigma_1 \frac{d\xi}{dt} + \sigma_2 v_{rel} \quad (3)$$

where σ_0 represents the stiffness of the bristles' spring, σ_1 is a damping coefficient and σ_2 is the viscous friction coefficient and $\dot{\xi} = \frac{d\xi}{dt}$.

In the Lu-Gre parametrization one provides $\xi_{ss}(v)$ as follows:

$$\xi_{ss}(v) = \frac{\text{sign}(v)}{\sigma_0} \left(f_c + (f_s - f_c) e^{-(v/v_s)^2}\right) \quad (4)$$

where f_c is the magnitude of Coulomb friction force, f_s is the magnitude of the static friction (*stiction*) force and v_s is the Stribeck

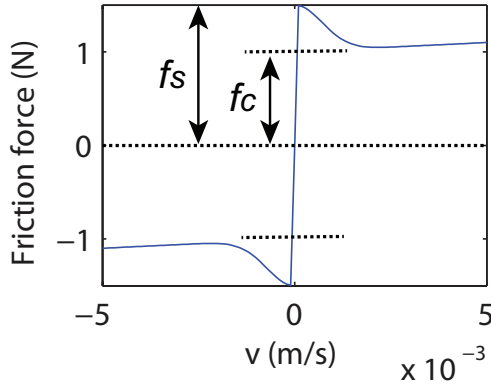


Figure 4: Typical steady state friction force versus velocity.

velocity, which controls the characteristics of the Stribeck effect where friction continuously decreases as relative velocity increases in the low velocity regime. A typical plot of the steady state friction force versus velocity is shown in Figure 4.

The bristle model is sufficiently general to capture most of the phenomena associated to friction. In particular, due to the dependency of the force on the relative velocity of the contact surfaces, the model is able to produce stick-slip motion, continuously switching from static to kinetic friction according to velocity regimes.

4. DIGITAL WAVEGUIDE SIMULATION OF FRET-STRING INTERACTION

In this section we consider the friction model reviewed in Section 3.1 to simulate the behavior of the string pressed against the fret in the vertical z direction but free to move in the horizontal y direction. We will first derive the continuous time system describing the string-fret interaction and then provide a discrete version of the model based on bilinear transform. Furthermore, we provide a scheme to compute the solution of the nonlinear difference equation describing the string-fret node, which is based on the so-called K-method [15].

4.1. Continuous Time String-Fret Node

Let us denote by $u_y(x, t)$ and $u_z(x, t)$, respectively, the value of the string displacement at time t and position x along the string for the y and z polarization modes.

Disregarding nonlinear [16] and dispersive effects [17, 18], the wave equation holds for segments of the string not in contact with other objects such as the plectrum or the player's finger and the fret. Assume that the only object in contact with the string is the fret, touching the string on a segment of width Δ and centered at coordinate x_f , then for a string of length L_s we have

$$c^2 \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}; \quad x \in]0, x_f - \frac{\Delta}{2} [\cup] x_f + \frac{\Delta}{2}, L_s [, \quad (5)$$

where $u(x, t)$ denotes any of the two polarization displacement $u_y(x, t)$ or $u_z(x, t)$, while $c = \sqrt{K_0/\mu}$ is the propagation velocity, K_0 is the tension of the string, and μ is the linear mass density,

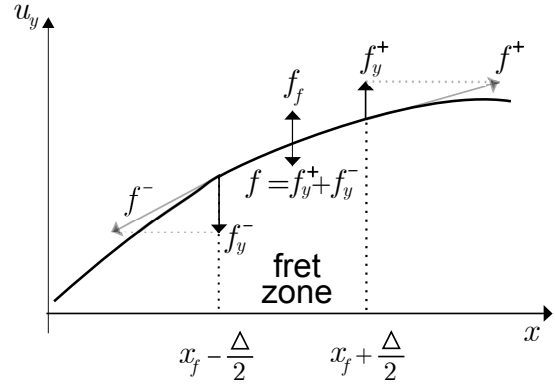


Figure 5: String segment subject to tensile forces f^- and f^+ and friction f_f against the fret. For the y polarization, the tensile forces are projected along the y direction, obtaining f_y^- and f_y^+ . The resultant $f = f_y^- + f_y^+$ of the projected tensile forces is considered as acting at the point x_f .

and they are all assumed to be constant. Here we have disregarded all propagation losses along the string, as these can be consolidated at one of the extremities and embedded in the bridge model [19].

The solution of (5) can be written in D'Alembert form as a superposition of a left-going u^- and a right-going u^+ wave:

$$u(x, t) = u^-(x, t) + u^+(x, t) = u_l(t + x/c) + u_r(t - x/c), \quad (6)$$

where $u_l(x/c) = u_r(x/c) = u(x, 0)/2$ for a static initial displacement condition.

For the vertical polarization mode u_z the portion of the string in contact with the fret can be largely assumed to be clamped in normal playing conditions. In this case the left-going wave u_z^- is perfectly reflected at the fret back towards the bridge, i.e., $u_z^+(x_f, t) \approx -u_z^-(x_f, t)$. As already remarked, the same is not true for the horizontal polarization mode u_y . On the string-fret contact segment, which we will also refer to as the fret zone shown in Figure 5, the equilibrium equation of the string with the bristle based dynamic system modeling friction (2) is enforced:

$$\mu \Delta \frac{\partial^2 u_y}{\partial t^2} = f(t) - f_f(\xi, \dot{\xi}, v_y) \quad (7)$$

$$x \in]x_f - \frac{\Delta}{2}, x_f + \frac{\Delta}{2} [,$$

where the force $f(t)$ is the resultant of the transversal component of the tensile force of the string acting at the extreme points of the contact segment and $f_f(\xi, \dot{\xi}, v_y)$ is the friction force (3). The velocity v_y is the relative velocity of the string over the fret, which coincides with string displacement velocity in the y -polarization mode. It is therefore convenient to rewrite (7) all in terms of $v_y(x, t) = \frac{\partial u_y}{\partial t}$:

$$\mu \Delta \frac{\partial v_y}{\partial t} = f(t) - f_f(\xi, \dot{\xi}, v_y) \quad (8)$$

$$x \in]x_f - \frac{\Delta}{2}, x_f + \frac{\Delta}{2} [.$$

At small string displacements, for the tensile force we have:

$$f(t) = K_0 \left(\frac{\partial u_y}{\partial x} \Big|_{x=x_f+\frac{\Delta}{2}} - \frac{\partial u_y}{\partial x} \Big|_{x=x_f-\frac{\Delta}{2}} \right). \quad (9)$$

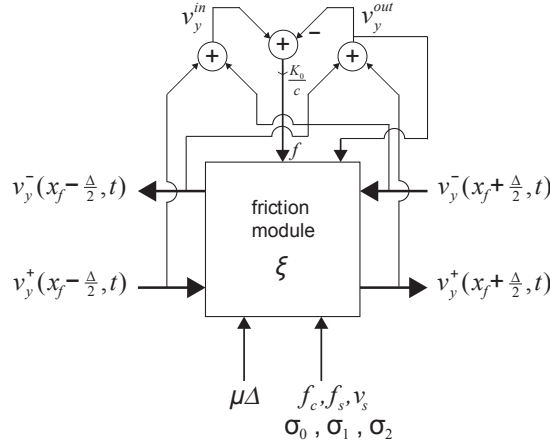


Figure 6: Block diagram representing the friction node for the simulation of string-fret interaction.

Since from (6) we have

$$\frac{\partial u}{\partial x} = \frac{1}{c} (v^-(x, t) - v^+(x, t)) \quad (10)$$

where

$$\begin{aligned} v^-(x, t) &= \frac{\partial u^-}{\partial t} \\ v^+(x, t) &= \frac{\partial u^+}{\partial t} \end{aligned} \quad (11)$$

then (9) can be rewritten as follows:

$$f(t) = \frac{K_0}{c} (v_y^{in}(t) - v_y^{out}(t)), \quad (12)$$

where we have defined v_y^{in} as the velocity wave entering the fret zone and v_y^{out} as the velocity wave leaving the fret zone, i.e.,

$$\begin{aligned} v_y^{in}(t) &= v_y^-(x_f + \frac{\Delta}{2}, t) + v_y^+(x_f - \frac{\Delta}{2}, t) \\ v_y^{out}(t) &= v_y^+(x_f + \frac{\Delta}{2}, t) + v_y^-(x_f - \frac{\Delta}{2}, t). \end{aligned} \quad (13)$$

Assimilating $v_y(x_f, t)$ to $v_y^{out}(t)$, i.e., shrinking the system (8) to a point, while retaining the finite mass $\mu\Delta$, we obtain the string-fret node equation:

$$\mu\Delta \frac{dv_y^{out}}{dt} = \frac{K_0}{c} (v_y^{in}(t) - v_y^{out}(t)) - \sigma_0 \xi - \sigma_1 \frac{d\xi}{dt} - \sigma_2 v_y^{out}(t), \quad (14)$$

where we have substituted (3) and (12) in (7) after establishing that $v_{rel} = v_y^{out}$. A block diagram of the string-fret interaction node is shown in Figure 6.

The bristle displacement function ξ in (14) must satisfy equation (2). Defining a state vector

$$\mathbf{x} = \begin{bmatrix} v_y^{out} \\ \xi \end{bmatrix}, \quad (15)$$

equations (14) and (2) can be put in the form of a nonlinear state space system:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}v_y^{in} + \mathbf{e}\phi \\ \phi = \rho(\mathbf{x}) \end{cases}, \quad (16)$$

where

$$\begin{aligned} \mathbf{A} &= \frac{-1}{\mu\Delta} \begin{bmatrix} \sigma_2 + \frac{K_0}{c} & \sigma_0 \\ 0 & 0 \end{bmatrix} \\ \mathbf{b} &= \frac{K_0}{c\mu\Delta} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \mathbf{e} &= \begin{bmatrix} -\frac{\sigma_1}{\mu\Delta} \\ 1 \end{bmatrix} \end{aligned} \quad (17)$$

and

$$\rho \left(\begin{bmatrix} v_y^{out} \\ \xi \end{bmatrix} \right) = v_y^{out} \left(1 - \alpha(v_y^{out}, \xi) \frac{\xi}{\xi_{ss}(v_y^{out})} \right) \quad (18)$$

is a scalar function of the state vector.

In the form (16) the system describing the string-fret node is ready for suitable discretization required in digital simulations of strings.

4.2. Discrete Time Computation of the String-Fret Node

In this section we carry out the discretization of the system (16) using the bilinear transformation. This method has the advantage of preserving passivity of the system, which prevents the introduction of instability due to numerical approximation of the derivatives. We will also show how to handle the delay-free loops in the computation.

The system in (16) is characterized by first order derivatives. In Laplace transform a differentiator is equivalent to multiplication by the Laplace variable s . By bilinear transformation, s is replaced by $2(1 - z^{-1})/T(1 + z^{-1})$, where T is the sampling time interval. Accordingly, a first order differential equation of the type

$$\dot{\eta}(t) = f(\eta(t), t) \quad (19)$$

is led by bilinear transformation to the recurrence

$$\eta(n) = \eta(n-1) + \frac{T}{2} [f(\eta(n), n) + f(\eta(n-1), n-1)], \quad (20)$$

where we dropped the factor T in the arguments of the functions.

Using this rule, it is easy to discretize the system (16). The discrete version of equation for the first state component expresses the current value of the output velocity $v_y^{out}(n)$ in terms of past values of v_y^{out} , present and past values of the input velocity v_y^{in} , present and past values of ϕ and present and past values of bristle deflection ξ . The discrete version of the equation for the second component of the state becomes the recurrence:

$$\xi(n) = \xi(n-1) + \frac{T}{2} (\phi(n) + \phi(n-1)). \quad (21)$$

This recurrence can be substituted in the first state component recurrence in order to remove the dependency from the present value of ξ , obtaining

$$\begin{aligned} v_y^{out}(n) &= c_1 v_y^{out}(n-1) + c_2 \xi(n-1) \\ &+ c_3 [v_y^{in}(n) + v_y^{in}(n-1)] + c_4 [\phi(n) + \phi(n-1)] \end{aligned} \quad (22)$$

where

$$\begin{aligned} c_1 &= \frac{2\mu\Delta c - Tc\sigma_2 - TK_0}{2\mu\Delta c + Tc\sigma_2 + TK_0} \\ c_2 &= -\frac{2Tc\sigma_0}{2\mu\Delta c + Tc\sigma_2 + TK_0} \\ c_3 &= \frac{TK_0}{2\mu\Delta c + Tc\sigma_2 + TK_0} \\ c_4 &= -\frac{Tc(\sigma_1 + \frac{T}{2}\sigma_0)}{2\mu\Delta c + Tc\sigma_2 + TK_0}. \end{aligned} \quad (23)$$

We are then left with a recurrence for v_y^{out} that depends on known values, except for that of $\phi(n)$. Yet, the equation for ϕ requires the value of $v_y^{out}(n)$ in order to be computed, which is a delay-free loop of the system. This delay-free loop must be properly handled in order to be able to find the solution, as described next.

Substituting the recurrence for $v_y^{out}(n)$ and that for $\xi(n)$ in the vector argument of the function ρ in (16), one obtains an equation of the type

$$\phi(n) = g(\phi(n), n), \quad (24)$$

where g is a known function, which is built from ρ by isolating the dependency on ϕ and reducing all other dependencies to an explicit dependency on time index n . This equation can be solved by finding, at any sample index n , a local zero of the function

$$\zeta - g(\zeta, n), \quad (25)$$

which can be achieved by means of Newton-Raphson root finding method. Look-up tables for the roots can be precalculated in order to ease real-time computation [15]. The root ζ of (25) is assigned to $\phi(n)$ and all other quantities are known in order to compute $v_y^{out}(n)$ and $\xi(n)$, which describes how to handle the delay-free loop in the computation.

4.3. Fret Junction in Digital Waveguides

The discrete time realization of the fret-string interaction block illustrated in the previous section is directly usable as a block in digital waveguides for the synthesis of strings based on velocity waves. The block is only included in the waveguide simulating the horizontal y -polarization mode. The input velocity v_y^{in} is obtained by summing the input velocities v_{in}^+ and v_{in}^- from the two rails of the waveguide. The output velocity v_y^{out} obtained from the fret-string system is equally fed to the two rails of the waveguide. In order to force the output velocity at the fret contact point, a scattering junction of the type

$$\begin{bmatrix} v_{out}^- \\ v_{out}^+ \end{bmatrix} = S_c \begin{bmatrix} v_{in}^- \\ v_{in}^+ \end{bmatrix} + \frac{v_y^{out}}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (26)$$

where

$$S_c = \frac{1}{2} \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} \quad (27)$$

is included, similar to what described in [6] in order to force after-collision displacement.

As this paper is part of a larger project for the accurate simulation of the guitar, and as in our system displacement waves are preferred for their ease of use in the detection of string-neck or string-fret collisions, differentiator and integrator blocks have to be introduced in order to obtain the input velocity. These blocks can be realized, respectively, by directly taking the first order difference of the incoming signal and by a discrete time leaky integrator.

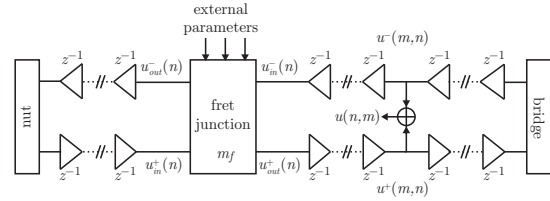


Figure 7: Inclusion of a fret junction in a digital waveguides for string displacement waves (horizontal polarization).

A valid alternative is to use a differentiator and an integrator derived by applying the bilinear transformation to the analog differentiator and integrator, similar to what described in Section 4.2. The bilinear differentiator is given by the following recurrence:

$$v(n) = -v(n-1) + \frac{2}{T} (u(n) - u(n-1)) \quad (28)$$

while the bilinear integrator is

$$u(n) = u(n-1) + \frac{T}{2} (v(n) + v(n-1)) \quad (29)$$

as in (20). Bilinear integrator and differentiator also have the advantage of being inverse of each other.

The insertion of the fret junction in a displacement wave based digital waveguide is shown in Figure 7.

The parameters of the underlying friction model are the magnitudes f_c of the Coulomb and f_s of the stiction force, the Stribeck velocity v_s , together with bristles' stiffness σ_0 , damping σ_1 and viscous friction coefficient σ_2 . Also, the elasto-plastic map function $\alpha(v_{rel}, \xi)$ needs to be specified. In first approximation we disregarded elasto-plastic phenomena and enforced a simplified Lu-Gre model setting $\alpha(v_{rel}, \xi) = 1$.

The string-fret friction parameters can and should be measured accurately from the string-fret friction characteristics. String-fret friction measurement will be the object of further studies. A special laboratory set up is required in which a free piece of guitar string is pulled, at several constant velocities, over a single fret. The friction force is measured by means of a miniature accelerometer.

In our preliminary experiments we used reference values for these parameters as follows. For the σ parameters we let $\sigma_0 = 10^5$ N/m, $\sigma_1 = 300$ Ns/m and $\sigma_2 = 10^{-3}$ Ns/m. For the stiction force we used a 50% increment of the Coulomb force level, i.e., $f_s = 1.5 \times f_c$, where forces are measured in Newtons N .

The Coulomb force can be estimated as the the friction coefficient for metal, about 0.5, times the normal force at the fret. However, we found that the value of the friction coefficient for metals is too high for the simulation of the string. This is due to the fact that both string and fret are rounded and smooth surfaces, more resembling ball bearing than flat surfaces in contact. Indeed the string is also allowed to roll over the fret to some extent, generating torsional effects on the string. Large friction coefficients tend to stop the string and / or introduce noise that is not typical of this type of interaction.

The normal force at the fret can be estimated from the vertical component of the tensile force due to the bending of the string at the fret, given by the force $F_{1,z}$ as shown in Figure 8. This force is essentially given by the slope of the string at the fret times the

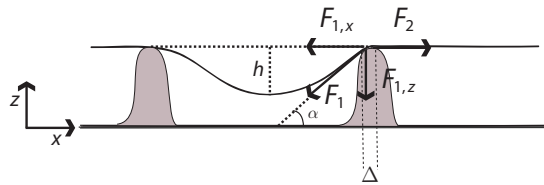


Figure 8: Components of the tensile force due to the bending of the string by the player's finger near the fret.

string tension K_0 . It consists of two components: a static component due to the bending of the string by the finger and a dynamic one due to string motion in the vertical z polarization. The dynamic component introduces non-linear coupling of the z and y polarizations. However, in usual playing modes, all dynamic variations can be disregarded as they only provide a very minor variation of the normal force due to the largely bent string at the fret.

Finally, for the Stribeck velocity we used a reference value of $v_s = 10^{-3} \text{ m/s}$.

The fret junction must be completed by a model of the player's finger placed next to the fret. An accurate model can be derived from the damped spring-mass system presented in [7] for the finger plucking where, in the case of the finger over the fret, the coordinates of the fingers are static. However, the effect of the finger behind the fret has no dramatical influence on the sound. Thus, a simpler reflector with damping can be suitably employed in normal playing conditions. A further completion of the model requires the inclusion of a second fret junction corresponding to the trailing fret on which the string is resting. However, this is quite unnecessary provided that one suitably blocks the oscillations on the right portion of the string to propagate to the left portion of the string with respect to the fret.

The dynamic model of friction contributes to provide an accurate simulation of the deflection of the string over the fret in the horizontal polarization. The string simulation is completed by a tension modulation module that allows us to obtain the pitch-bending characteristic of plucked guitar tones, which is otherwise absent in the wave equation as one assumes there that the tension is constant. Alternate implementations of tension modulation methods can be found in [20, 21].

5. CONCLUSIONS

In this paper we have considered the sliding of the string over the fret as a phenomenon to be modeled for the accurate synthesis of fretted string instruments. We provided a discrete model derived from bristle based models of friction. The continuous time model is described by a nonlinear state-space system, which is discretized by means of the bilinear transformation. The computation requires the solution of a nonlinear equation, which can be achieved by Newton-Raphson root finding method.

The acoustical results are very realistic and require very moderate amount of friction, as controlled by the friction coefficient in the Coulomb force, which is a parameter of the model. High friction coefficients tend otherwise to stop the string too early in the vertical polarization and introduce unnatural noise.

Sound examples can be found at <http://staffwww.itn.liu.se/~giaev/soundexamples.html>.

6. ACKNOWLEDGMENTS

The author wishes to thank Balázs Bank for suggesting the problem and for fruitful discussions.

7. REFERENCES

- [1] J. Woodhouse, "On the synthesis of guitar plucks," *Acta Acustica*, vol. 90, pp. 928–944, 2004.
- [2] F. Eckerholm and G. Evangelista, "The PluckSynth touch string," in *Proc. of Digital Audio Effects Conf. (DAFx '08)*, Helsinki, Finland, Sept. 2008, pp. 213–220.
- [3] G. Evangelista, "Modified phase vocoder scheme for dynamic frequency warping," in *Proc. of IEEE 3rd International Symposium on Communications, Control and Signal Processing (ISCCSP 2008)*, St. Julians, Malta, March 2008, pp. 1291–1296.
- [4] F. Germain and G. Evangelista, "Synthesis of guitar by digital waveguides: Modeling the plectrum in the physical interaction of the player with the instrument," in *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA-09)*, 2009.
- [5] G. Evangelista, "Physically inspired playable models of guitar, a tutorial," in *Proc. 4th International Symposium on Digital Communications, Control and Signal Processing (ISCCSP)*, 2010, pp. 1–4.
- [6] G. Evangelista and F. Eckerholm, "Player-instrument interaction models for digital waveguide synthesis of guitar: Touch and collisions," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 822–832, 2010, Special issue on Virtual Analog Audio Effects and Musical Instruments.
- [7] G. Evangelista and J. O. Smith III, "Structurally Passive Scattering Element for Modeling Guitar Pluck Action," in *Proc. of Digital Audio Effect Conf. (DAFx'10)*, 2010, pp. 10–17.
- [8] C. Canudas de Wit, H. Olsson, K. J. Åström, and P. Lischinsky, "A new model for control of systems with friction," *IEEE Trans. Automat. Contr.*, vol. 40, no. 3, pp. 419–425, 1995.
- [9] J. Swevers, F. Al-Bender, C. G. Ganseman, and T. Prajogo, "An integrated friction model structure with improved presliding behavior for accurate friction compensation," *IEEE Trans. Automat. Contr.*, vol. 45, no. 4, pp. 675, Apr. 2000.
- [10] Neville H. Fletcher and Thomas D. Rossing, *The Physics of Musical Instruments*, Springer, New York, 2nd edition, 1998.
- [11] G. U. Varieschi and C. M. Gower, "Intonation and compensation of fretted string instruments," *American Journal of Physics*, vol. 78, no. 1, pp. 47–55, January 2010.
- [12] P. Dupont, V. Hayward, B. Armstrong, and F. Altpeter, "Single state elasto-plastic friction models," *IEEE Trans. Automat. Contr.*, vol. 47, no. 5, pp. 787–792, 2002.
- [13] S. Serafin, F. Avanzini, and D. Rocchesso, "Bowed string simulation using an elasto-plastic friction model," in *Proc. Stockholm Music Acoustics Conf. (SMAC 2003)*, Stockholm, Sweden, Aug. 2003, pp. 95–98.

- [14] F. Avanzini, S. Serafin, and D. Rocchesso, “Interactive simulation of rigid body interaction with friction induced sound generation,” *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 5, Part 2, pp. 1073–1081, Sept. 2005.
- [15] G. Borin, G. De Poli, and D. Rocchesso, “Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems,” *IEEE Trans. Speech and Audio Processing*, vol. 8, no. 5, pp. 597–606, 2000.
- [16] D. W. Oplinger, “Frequency response of a nonlinear stretched string,” *J. Acoust. Soc. Amer.*, vol. 32, pp. 1529–1538, 1960.
- [17] G. Evangelista and S. Cavaliere, “Real-time and efficient algorithms for frequency warping based on local approximations of warping operators,” in *Proc. of Digital Audio Effects Conf. (DAFx ‘07)*, Bordeaux, France, Sept. 2007, pp. 269–276.
- [18] I. Testa, G. Evangelista, and S. Cavaliere, “Physically Inspired Models for the Synthesis of Stiff Strings with Dispersive Waveguides,” *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 7, pp. 964–977, July 2004, special issue on Model-Based Sound Synthesis.
- [19] J. O. Smith III, “Efficient synthesis of stringed musical instruments,” in *Proc. of the International Computer Music Conference*, Tokyo, Japan, 1993, pp. 64–71.
- [20] B. Bank, “Energy-based synthesis of tension modulation in strings,” in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 2009, pp. 365–372.
- [21] T. Tolonen, V. Välimäki, and M. Karjalainen, “Modeling of tension modulation nonlinearity in plucked strings,” *IEEE Trans. Speech and Audio Process.*, vol. 8, no. 3, pp. 300–310, May 2000.

A HIGH-RATE DATA HIDING TECHNIQUE FOR AUDIO SIGNALS BASED ON INTMDCT QUANTIZATION

Jonathan Pinel, Laurent Girin,*

GIPSA-Lab, Grenoble Institute of Technology,
Grenoble, France

{jonathan.pinel, laurent.girin}@gipsa-lab.grenoble-inp.fr

ABSTRACT

Data hiding consists in hiding/embedding binary information within a signal in an imperceptible way. In this study we propose a high-rate data hiding technique suitable for uncompressed audio signals (PCM as used in Audio-CD and .wav format). This technique is appropriate for non-secrity applications, such as enriched-content applications, that require a large bitrate but no particular robustness to attacks. The proposed system is based on a quantization technique, the Quantization Index Modulation (QIM) applied on the Integer Modified Discrete Cosine Transform (IntMDCT) coefficients of the signal and guided by a PsychoAcoustic Model (PAM). This technique enables embedding bitrates up to 300 kbps (per channel), outperforming a previous version based on regular MDCT.

1. INTRODUCTION

Data hiding consists in imperceptibly embedding information in a media. Theoretical foundations can be found in [1], and the first papers and applications dedicated to audio signals were developed in the 90's (e.g. [2, 3]). The main use (and probably original use) of data hiding for audio signals is the Digital Rights Management (DRM): the embedded data are usually copyrights or information about the author or the owner of the media content (in this case data hiding is referred to as *watermarking*). For such applications, the size of the embedded data is usually small, and a crucial issue is the robustness of the watermark to malicious attacks. Therefore, researches have long focused on enhancing the security and robustness of the data hiding techniques, at the price of limited embedding bitrate.

Data hiding is now used for non-secrity applications as well (e.g. [4]). In this paper we focus on "enriched-content" applications where data hiding is used to transmit side-information to the user, in order to provide additional interaction with the media. In this context, the specifications of data hiding are different from security applications (and somewhat opposed): here, a high embedding bitrate is generally required to provide substantial interactive features. Therefore, the technical issue is usually to maximize the embedding bitrate under the double constraint of imperceptibility and robustness. However, in contrast to security applications, robustness is here of a lesser importance because the user has no reason to impair the embedded data.

In this paper, we focus on high-rate data hiding for uncompressed audio signals (e.g. 44.1kHz 16-bit PCM samples, such as audio-CD, .wav, .aiff, .flac formats), with potential application to

enriched-content musical processing. For example, the so-called Informed Source Separation techniques developed in [5, 6] use embedded data to ease the separation of the different musical instruments and voices that form a music signal. In the present study, the embedding constraints are inaudibility and compliance with uncompressed format (16-bit time-domain PCM).

The system presented here is an improved version of the system previously presented in [7]. As in [7], it is a quantization-based embedding scheme, the quantization technique being the Quantization Index Modulation (QIM) applied on the Time-Frequency (TF) coefficients of the signal, and the computation of the embedding capacities is guided by a PsychoAcoustic Model (PAM) to ensure inaudibility. However, the TF transform used in the present study is the IntMDCT (Integer Modified Discrete Cosine Transform, [8]), which is an integer approximation of the MDCT [9] used in our previous study [7]. The interest of this transform is to provide directly the embedded signal in the PCM format. For the same reason, it was used in [10]. However, in [10] the PAM and the capacities have to be recomputed at the decoder using the *lead bits* principle. In the present study we keep the two-step embedding process of [7] that avoids such recomputation. Altogether, the combination of such two-step embedding process with the IntMDCT yields a significant gain for the embedding bitrate.

The paper is organized as follows: Section 2 is a general overview of the system while Section 3 presents in more details the core blocks of the system. Section 4 shows experiments and results, and finally conclusions and perspectives are discussed in Section 5.

2. GENERAL OVERVIEW OF THE SYSTEM

In this section we present the main principles of the data hiding system. The functional blocks will be detailed in the next section. The system consists of two main blocks (see Fig. 1): an *embedder* used to embed the data into the host signal x , and a *decoder* used to recover the data from the embedded host signal x^w ; the decoder is "blind" in the sense that the original signal is assumed to be unknown from the decoding part.

2.1. Embedding

The embedding is performed in the Time-Frequency (TF) domain. Therefore—at the embedder—the time-domain input signal x is first transformed in the time-frequency (TF) plan (Block ①). Instead of using the MDCT (as in [7]), the transform used here is its integer approximation, the IntMDCT [8]. The embedding process consists in quantizing the IntMDCT coefficients $X(t, f)$ (Block ④

* This work was supported by the French National Research Agency (ANR) in the context of the DReaM project (ANR 09 CORD 006).

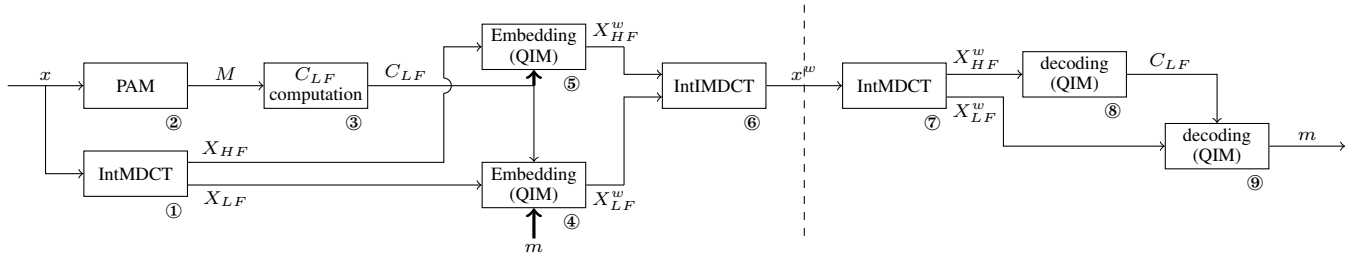


Figure 1: Block diagram of the system. Left of the dashed line is the embedder, right is the decoder. LF (resp. HF) stands for low frequency (resp. high frequency) and w denotes vector carrying embedded data.

and ⑤) using specific sets of quantizers $\mathcal{S}_{C(t,f)}$, following the QIM technique described in [11] (see Section 3.2). Once the IntMDCT coefficients are embedded, the signal is transformed back in the time-domain using the integer inverse MDCT (IntIMDCT, Block ⑥). The resulting embedded signal has integer values. Thus there is no need to perform the time-domain 16-bit PCM quantization and take into account the effects of the resulting noise on the embedding performance, as opposed to what was done in [7] (see Section 3.1).

For each frame t and for each frequency bin f , the PAM (Block ②) provides a masking threshold $M(t, f)$ used to compute the embedding capacity $C(t, f)$ (Block ③), *i.e.* the maximum size of the binary code to be embedded in that TF bin under inaudibility constraint. The embedding capacity $C(t, f)$ determines at the same time *how much* information is embedded (at TF bin (t, f)) and *how* it is embedded and retrieved. Consequently, the set of capacity values $C(t, f)$ must be known at the decoder and they have either to be estimated from the transmitted signal at the decoder (as in [10]), or to be transmitted within the host signal x as a part of the embedded data themselves (as in [7]). We keep the line of [7] and propose the following process:

- At the embedder, after IntMDCT transform, the IntMDCT coefficients are separated into a “low-frequency” part (denoted $_{LF}$ on Fig. 1 and thereafter, accounting for 15/16 of the spectrum) and a “high-frequency” part (denoted $_{HF}$, accounting for 1/16 of the spectrum, see Section 3.4).
- Low frequencies are used to embed the “useful” side-information m that is to be transmitted within the host audio signal x . For this aim, the capacities $C_{LF}(t, f)$ are maximized under inaudibility. This is the core of the proposed method that will be described into details in Section 3.4.
- Then, high frequencies are used to embed the values of the resulting capacities $C_{LF}(t, f)$ which totally configure the data hiding process in the low-frequency region. To do this, the values of $C_{HF}(t, f)$ must be known at both the embedder and the decoder. Hence they are set to fixed values (*i.e.* independent of frame index and signal content), exploiting the fact that in the highest frequency region the human hearing system is quite inefficient. Because the high-frequency capacities do not depend on t , they are denoted $C_{HF}(f)$.

2.2. Decoding

The decoding process somehow consists of the reverse operations: the embedded signal x^w is first transformed in the TF domain (Block ⑦) and the resulting IntMDCT coefficients are separated

into high and low frequencies subvectors, similarly to the embedder. As the capacities $C_{HF}(f)$ are known to the decoder, the information embedded in the high-frequency region is first extracted (Block ⑧), resulting in decoded $C_{LF}(t, f)$ values. This latter information is then used to decode the “useful” information m embedded in the low-frequency region (Block ⑨).

Note that if the synchronization is not treated in this paper, at least several basic schemes are usable, like for example checksums as used in [10].

3. DETAILED PRESENTATION

3.1. Time-frequency transform

The choice of the MDCT in [7] was mainly guided by the fact that it is a TDAC (Time Domain Aliasing Cancellation) transform. It also has the perfect reconstruction property, it is critically sampled and its coefficients are real, which enables an easy use of quantization techniques. In the present study, we use the integer approximation of the MDCT, the IntMDCT, in order to get rid of the noise introduced on the MDCT coefficients by the time-domain 16-bit PCM quantization [7]. We use a frame length of 2048 to have a sufficient frequency resolution while fitting music signals dynamic.

The principle of the integer approximation is to decompose the MDCT matrix in a product of matrices that are either permutation matrices or block diagonal 2×2 Givens Rotations matrices. The permutation matrices and their inverses maps directly from integer to integer and the 2×2 Givens Rotations can be approximated using the *Lifting Scheme* (see for example [8] for a detailed explanation).

3.2. Embedding technique

The Quantization Index Modulation (QIM) is a quantization-based embedding technique introduced in [11]. The scalar version of the technique is used here¹, which means that each IntMDCT coefficient $X(t, f)$ is embedded independently from the others.

The embedding principle is the following. If $X(t, f)$ is the IntMDCT coefficient at TF bin (t, f) that has to be embedded with $C(t, f)$ bits, then a unique set $\mathcal{S}_{C(t,f)}$ of $2^{C(t,f)}$ quantizers $\{Q_c\}_{0 \leq c \leq 2^{C(t,f)}-1}$ is defined with a fixed arbitrary rule. This implies that for a given value $C(t, f)$ the set generated at the decoder is the same as the one generated at the embedder. The quantization levels of the different quantizers are intertwined (see Fig. 2) and

¹ Note that in this particular case the technique is similar to the *improved LSB* embedding scheme.

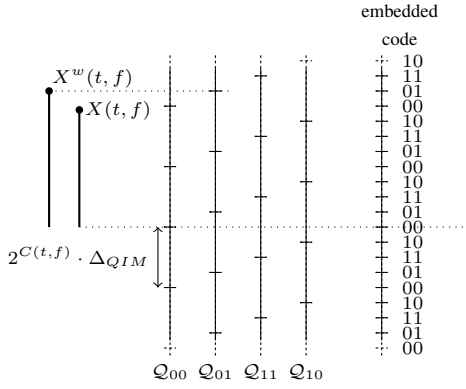


Figure 2: Example of QIM using a set $\mathcal{S}_{C(t, f)}$ of quantizers for $C(t, f) = 2$ with their respective gray code index and resulting global grid. The binary code 01 is embedded into the IntMDCT coefficient $X(t, f)$ by quantizing it to $X^w(t, f)$ using the quantizer indexed by 01.

each quantizer is indexed by a $C(t, f)$ -bit codeword c . Because the quantizers are regularly intertwined and because the IntMDCT coefficients are integer-valued, the quantization step of each quantizer is given by:

$$\Delta(t, f) = 2^{C(t, f)}. \quad (1)$$

Embedding the codeword c into the IntMDCT coefficient $X(t, f)$ is simply done by quantizing $X(t, f)$ with the quantizer Q_c indexed by c (see Fig. 2 for an example). In other words, the IntMDCT coefficient $X(t, f)$ is replaced with its closest code-indexed quantized value $X^w(t, f)$:

$$X^w(t, f) = Q_c(X(t, f)). \quad (2)$$

At the decoder, the set of quantizers $\mathcal{S}_{C(t, f)}$ is generated (and is the same as the one generated at the embedder) using the $C(t, f)$ decoded values in low-frequency and fixed values in high-frequency. Then, the quantizer Q_c with a level corresponding to the received embedded coefficient $X^w(t, f)$ is selected, and the decoded message is the index c of the selected quantizer. As the IntMDCT directly yields a PCM signal (due to the integer-to-integer mapping of the IntMDCT and IntIMDCT), there is no noise introduced by the conversion (in contrast to [7] when using the MDCT).

Obviously if one wants to transmit a large binary message, this message has to be previously split and spread across the different IntMDCT coefficients according to the local capacity values, so that each coefficient carries a small part of the complete message. Conversely, the decoded elementary messages have to be concatenated to recover the complete message.

3.3. Psychoacoustic model

The PAM used in our system (Block ②) is directly inspired from the PAM of the MPEG-AAC standard [12]. The output of the PAM is a masking threshold $M(t, f)$, which represents the maximum power of the quantization error that can be introduced while ensuring inaudibility. The calculations are made in the time-frequency domain, however the transform used for the PAM computations is not the IntMDCT but the FFT. The main computations consist in

a convolution of the FFT power spectrum of the host signal with a spreading function that models elementary frequency masking phenomena, to obtain a first masking curve. This curve is then adjusted according to the tonality of the signal, and the absolute threshold of hearing is integrated. After that, some pre-echo control is applied, resulting in the FFT masking threshold. The pre-echo control implemented is quite simple and only consists in taking the minimum of the computed masking threshold and the previous frame masking threshold multiplied by a constant $K > 1$. Taking a value close to 1 will yield a good pre-echo control but will limit the PAM efficiency (in term of embedding rate), while taking too big a value will lead to a poor pre-echo control (in this study $K = 2$). From the FFT spectrum and FFT masking threshold a signal-to-mask ratio (SMR) is computed (for each frequency bin f), and this SMR is then used to obtain the IntMDCT masking threshold $M(t, f)$ (by simply computing the ratio between the IntMDCT power spectrum coefficients and the SMR coefficients). This masking threshold $M(t, f)$ is then used to shape the embedding noise (under this curve), so that it remains inaudible. The masking threshold can also be translated by a factor of α dB so that the total payload matches exactly the size of the signal to be embedded m .

3.4. Capacities computation

The computation of the capacities $C(t, f)$ is the core of the proposed method. As the compliance to the PCM format is already ensured by the use of the IntMDCT, the problem is to optimize the embedding bitrate under inaudibility constraint. In the present study, this constraint is that the power of the embedding error in the worst case remains under the masking threshold $M(t, f)$ provided by the PAM. As the embedding is performed by uniform quantization, the embedding error in the worst case is equal to half the quantization step $\Delta(t, f)$, which is directly related to $C(t, f)$ through (1). The inaudibility constraint in a given TF bin can thus be written as:

$$\left(\frac{\Delta(t, f)}{2}\right)^2 < M(t, f). \quad (3)$$

For the low-frequency region of a given frame t , we simply combine (1) and (3) to obtain:

$$C_{LF}(t, f) < \frac{1}{2} \log_2(M(t, f)) + 1. \quad (4)$$

Since the capacity per coefficient is an integer number of bits, and we want to maximize this capacity, we choose:

$$C_{LF}(t, f) = \left\lfloor \frac{1}{2} \log_2(M(t, f)) + 1 \right\rfloor. \quad (5)$$

where $\lfloor \cdot \rfloor$ denotes the *floor* (rounding down) function. Experimentally, the resulting values are always lower than 15. Thus we can code those values with 4-bit codewords (from 0 to 15). However, embedding the high-frequency region with as many 4-bit codewords as there are frequency bins in the low-frequency zone is not achievable. For this reason, *embedding subbands* are defined as groups of adjacent frequency bins where the capacities $C(t, f)$ are fixed to the same value. The capacity value within each subband is given by applying (5) using the minimum value of the mask within the subband. In order to respect the inaudibility constraint in the high-frequency region, the capacities $C_{HF}(f)$ are fixed to 1 or 2

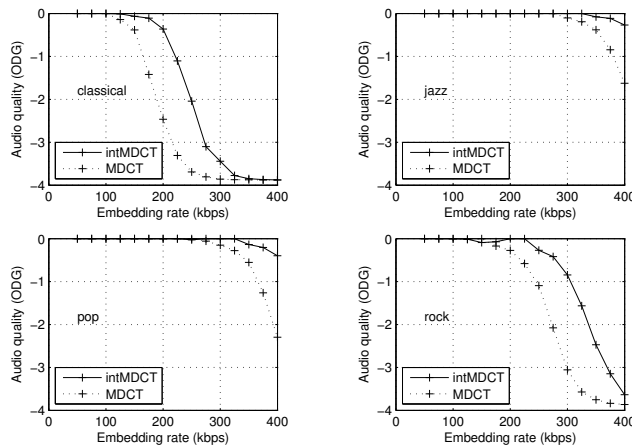


Figure 3: Audio quality as a function of embedding bitrate for a set of tracks used for the tests.

bits. In the present study the 1024 IntMDCT coefficients of each frame are split into 32 bands of 32 coefficients, and the last 2 sub-bands form the high-frequency zone.

4. RESULTS

The performance of the proposed data hiding system is evaluated in terms of audio quality of the embedded signal as a function of the embedding rate. The audio quality is estimated using the Perceptual Evaluation of Audio Quality (PEAQ) algorithm [13] and double-checked by informal listening tests. The PEAQ algorithm compares the embedded signal with the original signal, and provides a comparative score, called Objective Difference Grade (ODG). Grades range from 0 for inaudible effect to -4 for severe degradation. The tests were performed on twelve 10-second excerpts of 44.1-kHz 16-bit musical signals of different musical styles (classic, jazz, rock, pop. . .).

Fig. 3 shows some results and as we can see, the embedding bitrate is quite dependent of the audio content—as was already the case in [7]—thanks to (or due to) the PAM. When performing a comparison with the previous system [7], for the same ODG the embedding bitrate is quite higher for the new version (by about 50 kbps). It is also significantly higher than the 140 kbps announced in [10]. In particular, while maintaining inaudibility of the embedded data, bitrates up to 300 kbps can be reached for some very energetic signals (like pop music or jazz-rock). For less energetic signals (classical music) bitrates about 200 kbps are obtained. We can also see that in many cases the embedded data are inaudible with the system of the present study while it is not the case with the previous system of [7]. For most listeners, this is the case for example for an embedding rate of 375 kbps for the jazz track used in Fig.3.

5. CONCLUSION AND PERSPECTIVES

In this paper we presented a data-hiding technique for uncompressed audio signals that yields embedding bitrates of up to 300 kbps per channel for 44.1-kHz 16-bit music signals (depending on audio content). This represents more than 40% of a channel original rate and a significant gain over previous results obtained in [7]

and [10]. This technique can be used for “enriched-content” applications, as for instance the informed source separation system presented in [5, 6].

As compressed signals are now widely used, in future works we plan to look at the joint compression and watermarking problem by adapting the principles presented in this paper to compressed signals, for instance the scalable MPEG4-SLS format which also uses the IntMDCT.

6. REFERENCES

- [1] M. Costa. Writing on dirty paper. *IEEE Trans. Inform. Theory*, 29(3):439–441, 1983.
- [2] L. Boney, T. Ahmed, and H. Khaled. Digital watermarks for audio signals. In *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, Hiroshima, Japan, 1996.
- [3] I.J. Cox, M.L. Miller, and A.L. McKellips. Watermarking as communications with side information. *Proc. IEEE*, 87(7):1127–1141, 1999.
- [4] B. Chen and C.-E.W. Sundberg. Digital audio broadcasting in the FM band by means of contiguous band insertion and precanceling techniques. *IEEE Trans. Commun.*, 48(10):1634–1637, 2000.
- [5] M. Parvaix, L. Girin, and J.-M. Brossier. A watermarking-based method for single-channel audio source separation. In *Proc. IEEE Int. Conf. Acoust. and Speech, Signal Proc.*, Taipei, Taiwan, 2009.
- [6] M. Parvaix and L. Girin. Informed source separation of underdetermined instantaneous stereo mixtures using source index embedding. In *Proc. IEEE Int. Conf. Acoust. and Speech, Signal Proc.*, Dallas, Texas, 2010.
- [7] J. Pinel, L. Girin, and C. Baras. A high-capacity watermarking technique for audio signals based on mdct-domain quantization. In *Proc. Int. Congress on Acoustics*, Sydney, Australia, 2010.
- [8] R. Geiger, Y. Yokotani, and G. Schuller. Improved integer transforms for lossless audio coding. In *Proc. Asilomar Conf. Signal, Systems and Computers*, Pacific Grove, California, 2003.
- [9] J.P. Princen and A.B. Bradley. Analysis/synthesis filter bank design based on time domain aliasing cancellation. *IEEE Trans. Acoust. and Speech, Signal Proc.*, 64(5):1153–1161, 1986.
- [10] R. Geiger, Y. Yokotani, and G. Schuller. Audio data hiding with high data rates based on intMDCT. In *Proc. IEEE Int. Conf. Acoust. and Speech, Signal Proc.*, Toulouse, France, 2006.
- [11] B. Chen and G. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Trans. Inform. Theory*, 47(4):1423–1443, 2001.
- [12] ISO/IEC JTC1/SC29/WG11 MPEG. Information technology—Generic coding of moving pictures and associated audio information—Part 7: Advanced Audio Coding (AAC), IS13818-7(E), 2004.
- [13] ITU-R. Method for objective measurements of perceived audio quality (PEAQ), Recommendation BS.1387-1, 2001.

MAPPING BLOWING PRESSURE AND SOUND FEATURES IN RECORDER PLAYING

Leny Vincelas, Francisco García, Alfonso Pérez

Music Technology Group
Universitat Pompeu Fabra, Barcelona (SPAIN)
leny.vincelas@gmail.com
pul.editions@gmail.com
alfonso.perez@upf.edu

Esteban Maestre

Music Technology Group
Universitat Pompeu Fabra, Barcelona (SPAIN)
esteban.maestre@upf.edu
Center for Computer Research in Music and Acoustics
Stanford University, Stanford (USA)
esteban@ccrma.stanford.edu

ABSTRACT

This paper presents a data-driven approach to the construction of mapping models relating sound features and blowing pressure in recorder playing. Blowing pressure and sound feature data are synchronously obtained from real performance: blowing pressure is measured by means of a piezoelectric transducer inserted into the mouth piece of a modified recorder, while produced sound is acquired using a close-field microphone. Acquired sound is analyzed frame-by-frame, and features are extracted so that original sound can be reconstructed with enough fidelity. A multi-modal database of aligned blowing pressure and sound feature signals is constructed from real performance recordings designed to cover basic performance contexts. Out of the gathered data, two types of mapping models are constructed using artificial neural networks: (i) a model able to generate sound feature signals from blowing pressure signals, and therefore used to produce synthetic sound from recorded blowing pressure profiles via additive synthesis; and (ii) a model able to estimate the blowing pressure from extracted sound features.

1. INTRODUCTION

Studying and modeling instrumental music performance can be considered among the most challenging topics in the field of computer-aided analysis and generation of musical sound. In instrumental music practice, the performer transforms a musical score into control parameters used for driving sound production. Indeed, modeling how instrumental control signals are related to sound production appears as a pursuit with different application possibilities (e.g. performance, sound synthesis).

Regarding sound synthesis, the naturalness of synthetic sound greatly relies on how input controls are represented and mapped into sound features, both for the case of physical models or in spectral-domain sample-based frameworks. In general, physical models suffer from a lack of appropriate input control parameters while systems combining spectral modeling with pre-recorded samples do not explicitly consider instrumental gestures [1].

When aiming at low-intrusiveness measurement of instrumental gesture parameters, the idea of inferring instrumental control parameter signals from recorded sound appears as an attractive alternative. Indirect acquisition of instrumental gesture parameters in music playing represents a broader access to computationally studying music performance.

In this paper we present our recent work on sound-instrumental gesture mapping in recorder playing. Mapping models relating

sound features and blowing pressure are constructed from real data acquired in practice scenario, by means of a low-intrusiveness pressure acquisition system and a close-field microphone. On one hand a first mapping model able to generate sound features from blowing pressure is used to generate synthetic recorder sound from blowing pressure profiles using additive synthesis. On the other hand, a second mapping model is able to estimate the blowing pressure from features extracted from acquired recorder sound.

The rest of the paper remains as follows. Section 2 provides background and related work. In Section 3 we outline data acquisition, while Section 4 presents a spectral-domain representation used for analysis and re-synthesis of recorder sound. The construction and evaluation of mapping models is outlined in Section 5. Finally, Section 6 concludes by summarizing results and pointing out future directions.

2. BACKGROUND

Within the woodwind instrument family, the recorder can be considered as one of the simplest instruments, mainly due to the fact that its geometry fixes a number of input control parameters that for other instruments of the family are controllable by the performer (like for instance the distance between the mouth and the labium, which are not fixed for the transverse flute). Although the influence of the configuration of the mouth and the vocal tract has traditionally been a very controversial issue to which research efforts have been devoted [2], we assume here that only two control parameters enable the performer to achieve sound modulations during performance: the speed of the air jet (derived from the blowing pressure), and the fingering. Each fingering introduces distinct set of minima in the acoustic impedance that correspond to resonances. Blowing pressure, as opposed to fingering, presents a continuous nature while allowing the control of dynamics, timbre and fundamental frequency. By adjusting his blowing pressure, the performer is also able to select which impedance minimum controls the oscillation, and so can reach different notes while keeping the same fingering. In this work, we will focus on blowing pressure in the first oscillation mode assuming a fixed fingering.

During performance, blowing pressure is exponentially related to fundamental frequency [3], and linearly related to dynamics, as it has been shown also for the transverse flute [4, 5]. With regards to timbre in particular, Fletcher [4] pursued a study of the amplitude and of the harmonics as a function the intensity of pressure exerted by musicians, and showed two main phenomena (both related and expressed qualitatively): (i) the amplitudes of the second

and third harmonics, as related to the amplitude of the fundamental harmonic, increase with blowing pressure; and (ii) the high frequency content grows with blowing pressure. In this work we are aiming at providing model able to represent such relations in a flexible manner.

In general, there exist two main approaches for the acquisition of instrumental gesture parameters from real performance: direct acquisition and indirect acquisition. While direct acquisition methods are devised to measure the actual physical parameter from sensors conveniently placed in the instrument (and thus providing relatively straightforward and reliable measuring capability), indirect acquisition is based on the analysis of acquired sound signals and the extraction of parameters from which the actual instrumental control parameters used for produced such sound [6]. Given that intrusive measurement setups often keep performers from playing naturally for a prolonged period of time, the idea of being able to estimate instrumental gesture parameters (in this case blowing pressure) represents a motivation for our modeling approach to data-driven estimation of blowing pressure from sound features.

Conversely, it is also possible to find a mapping function that goes from control gestures to sound features. A first study [7] shows how statistical methods for non-linear prediction can be used for sound synthesis. A probabilistic inference technique (cluster-weighted modeling) is used to map performance controls of a bowed monochord to the amplitude of the first 25 harmonics of the sound.

A similar approach is used in [8] where a timbre model based on neural networks is presented. The neural networks are able to predict harmonic and residual spectral envelopes corresponding to violin bowing controls. The model is coupled to an additive synthesizer that fills the envelopes with harmonic and noisy content in order to produce synthetic violin sounds. This method has been successfully implemented for real time synthesis of violin sounds [9] where the spectral model is driven by bowing parameters captured using an ad-hoc control interface.

This approach for generating synthetic instrument sounds offers the possibility of substituting the habitual physical models by trained generative models, resulting particularly attractive for excitation-continuous musical instruments, such as bowed strings or wind instruments.

3. DATA ACQUISITION

Basically, data acquisition consisted in the synchronous acquisition of blowing pressure and sound signals from a real recorder practice scenario. A set of scripts was designed in order to cover a number of performance contexts when constructing a multi-modal database including aligned sound feature signals and blowing pressure signals. The scripts consisted on (i) a series of scripts covering different fingerings, articulations, note durations, and dynamics; and (ii) recording repetitions of a long crescendo (increasing blowing pressure) for each fingering. While sound was acquired by means of a close-field microphone attached to the body of the instrument, a special measurement system was designed for acquiring blowing pressure while maintaining a low impact on the performer's comfortability.

The mouthpiece block of an alto recorder was modified by enabling the connection of a pressure sensor without altering the timbre of the original instrument, as described in [10], leading to a significant reduction of the intrusiveness as compared to using a plastic catheter in the mouth of the musician [5, 11].

4. SOUND ANALYSIS

Acquired recorder sound is analyzed in the spectral domain with the aim of extracting a lower-dimensionality sound feature representation that enables (i) training a model able to learn mapping functions between extracted spectral parameters and blowing pressure, and (ii) to use such lower-dimensionality representation for reconstructing the original sound with enough fidelity so that those parameters can be used for synthesis purposes.

4.1. Some observations on the spectrum of the recorder sound

It was a good exercise for us to observe the time-varying spectra of a number of the crescendo recordings in the database, along with blowing pressure and fundamental frequency. Apart from the expected correlation between fundamental frequency and blowing pressure (as well observed in all the other recordings) [3], the overall amplitude of both harmonic and stochastic components increases with blowing pressure, having mid-range harmonic components to raise more prominently. Up to the expected change of oscillation mode starting to take place at higher pressure values (and in agreement with [12]), a differentiated behavior is observed for odd and even harmonic components, as it happens in other instruments of the same family [4]. This behavior leads to foresee a representation of odd and even harmonics independently. Therefore, we decided to represent each sound frame by the fundamental frequency, the noise floor, and the odd and even harmonic components. In this work we did not focus on higher modes of oscillations or chaotic, multiphonic transition states between oscillation modes.

4.2. Spectral-domain representation

In a first step, the audio signal is divided into overlapping frames and, for each of the frames, it is decomposed into deterministic (harmonic) and stochastic (residual) components via the spectral modeling synthesis (SMS) technique [13]. The algorithm implementation is based on a spectral peak-picking and tracking algorithm making use of the two-way mismatch (TWM) fundamental frequency estimation algorithm [14]. Given that notes have been segmented and fingerings (pitch values) have been annotated, octave errors are easily avoided by restricting the possible values of fundamental frequency candidates.

In the past, it has been shown that directly using harmonic amplitudes and frequencies in a timbre modeling pursuit leads to a highly complex and often inaccurate model [7], so with the idea of representing each frame by a reduced number of parameters, spectral envelope approximation is applied in a second step to each of the two extracted components. Given (i) our observations on the harmonic envelopes of recorded sound, (ii) the different modeling errors obtained during spectral envelope prediction (see next Section), and (iii) perceptual quality of obtained sound, it resulted best for our modeling and re-synthesis purposes to split the harmonic component into two different subcomponents: the odd harmonic envelope and the even harmonic envelope.

The next step is to approximate the spectral envelopes into consideration. Aiming at a compact representation of the harmonic spectral envelopes, harmonic peaks of each of the two sets (odd and even) are first interpolated by using a piecewise Hermite function. As for the stochastic spectral envelope (noise floor), a similar procedure is carried out starting from spectral peaks. Two example of the three envelopes are represented in Figure 1.

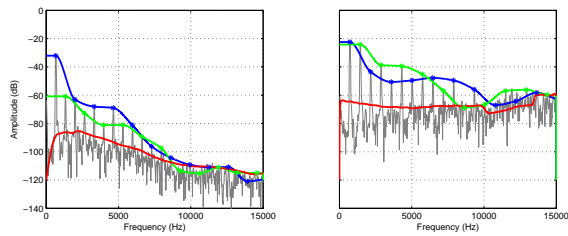


Figure 1: Spectral evolution of an E5 crescendo with its odd harmonic envelope (BLUE), even harmonic envelope (GREEN) and residual envelope (RED). Left: resulting spectrum for a blowing pressure of 197 Pa. Right: resulting spectrum for a blowing pressure of 1513 Pa.

Finally, each of the interpolated envelopes is encoded as a set of 15 mel-frequency cepstral coefficients (MFCCs), which for our experiments resulted to be a good compromise between size and fidelity. Out of the encoded three envelopes plus the value of fundamental frequency, it is possible to satisfactorily reconstruct the original sound via additive synthesis.

5. MAPPING MODELS

From encoded spectral envelopes and acquired blowing pressure signals, two frame-by-frame mapping functions are built for each fingering by means of artificial neural networks, using Matlab's neural network toolbox functions `newcf` and `newff`. The first model translates blowing pressure into sound features (spectral shape plus fundamental frequency), while the second one realizes the inverse mapping. Models are evaluated by means of 10-fold cross-validation. In both cases, separating harmonic envelopes into odd and even components led to obtain lower prediction errors; therefore our decision (also based on early observations by Fletcher [12]) to split harmonics and work from separated envelopes.

5.1. Mapping blowing pressure to sound features

A generative spectral model predicts, from the blowing pressure signal given in a frame-by-frame fashion, the three spectral envelopes (odd harmonics, even harmonics and residual) and fundamental frequency required to reconstruct sound. The prediction is carried out using two neural networks working in parallel. A first two-layer cascade-forward backpropagation network takes as input the blowing pressure and its derivative, and predicts the fundamental frequency plus the 15 MFCCs corresponding to each of the three spectral envelopes (2 inputs to 46 outputs). This network is trained with a subset of the training dataset which is composed of frames with harmonic structure. Since this model provides a coherent prediction for harmonic frames, an additional binary-output network (three-layer feed-forward backpropagation, 2 inputs to 1 output) capable of predicting harmonic state (whether a frame is voiced or not) was used as an activation to the first network. This last network was trained using the full dataset.

The results of spectral envelope prediction showed a highly robust and reliable performance, reaching a averaged correlation coefficient of 0.95 in the MFCC space. In Figure 2 it can be observed how predicted envelopes (dashed lines) almost match actual envelopes (continuous lines) for one frame of the signal. With respect to the prediction of fundamental frequency, in some cases we

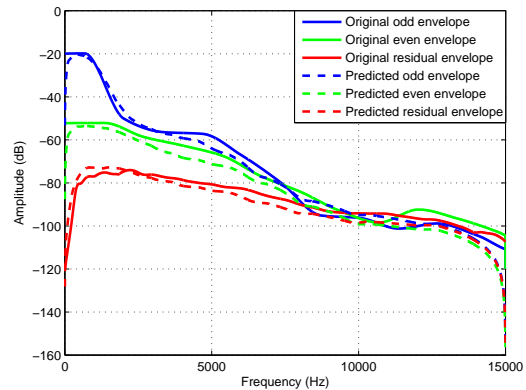


Figure 2: Example of spectral envelope prediction for a frame of sound in E5.

observed a maximum absolute error of 5Hz for some fingerings in the lower register. An averaged correlation coefficient of 0.83 was obtained when evaluating the performance of the second network. A preliminary additive synthesizer was implemented. It takes as input the note durations and fingerings from a segmented performance of a known score, together with the blowing pressure profiles from a previous recording of the same score. The synthesizer predicts envelopes and fundamental frequency and generates sound by means of additive synthesis: the harmonic sequence is generated from the fundamental frequency, partial amplitudes are set to the values in predicted harmonic spectral envelopes, and finally the residual component is generated as white noise filtered in the spectral domain by attending to the predicted residual spectral envelope. Although produced sound is perceived as very realistic, when subjectively comparing it with original sound, transients show in general a worse behaviour, mainly because only harmonic frames were used for the training. Example sounds can be listened on-line¹.

5.2. Estimating blowing pressure from sound features

The second model is able to estimate blowing pressure value from the MFCCs and fundamental frequency extracted for a given frame. A two-layer cascade-forward backpropagation network is trained with data from analyzed frames (46 inputs to 1 output) of each fingering in the database, leading to one different model per fingering. An averaged correlation coefficient of 0.91 was obtained for all fingerings. Figure 3 shows a prediction example: it is compared a recorded blowing pressure profile (red line) with the predicted signal (dashed blue line) of blowing pressure of one of the recordings, legato articulation. It can be observed that predicted signal suffers from fluctuations, however general shape is preserved and types of articulation can be identified.

6. CONCLUSION

We have presented the construction of mapping models relating sound features and blowing pressure in recorder playing. The

¹<http://ccrma.stanford.edu/~esteban/recsyntspec.html>

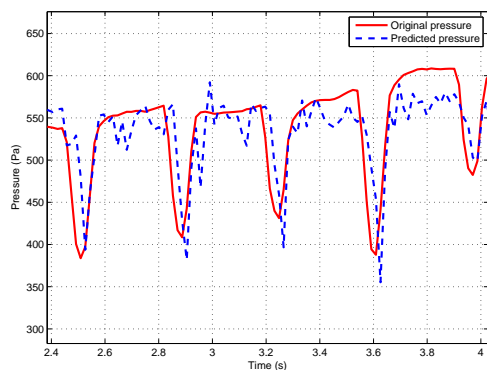


Figure 3: Comparison between the real blowing pressure (RED) and the predicted blowing pressure (BLUE) for E5.

models are built using artificial neural networks, which have been trained from blowing pressure and sound features extracted from real performance data. Blowing pressure is acquired by means of a low-intrusiveness pressure acquisition system based on a mechanical modification of the block of an alto recorder, to which a pressure sensor was attached in a low-invasive setup that allowed to play naturally. A first mapping model is trained to generate, given a blowing pressure signal, spectral envelopes (odd and even harmonic, plus noise floor) and fundamental frequency, so that sound can be faithfully synthesized via additive synthesis. A second model is able to estimate the blowing pressure from audio attributes extracted from acquired recorder sound. Both models showed a fairly good performance. For the case of sound feature prediction, an additive sound synthesizer was implemented as a proof-of-concept system able to generate realistic sound from blowing pressure and fingering information. Sound quality was evaluated only qualitatively by the authors and collaborators, but more formal subjective studies are to be carried out along with a number of improvements.

Results on spectral-domain sound generation from instrumental gesture controls shed light on the idea of using gesture-driven spectral models for enhancing sample-based sound synthesis, or even in real-time applications as in [9] where a Max/Msp allowed a real-time synthesis from a virtual instrument interface. The spectral envelope representation based on separating odd and even harmonics as two different envelopes showed to be a good approach for this instrument. Regarding Some improvements need to be carried out, especially for the case of transient modeling, for which the performed worse: time-domain signal analysis and synthesis from blowing pressure could result to be crucial, or even a hybrid model combining simple physical models (for transient parts) and spectral models (for steady-state parts). Another possibility would be to use this synthesis framework with automatically generated gesture parameter signals, as it was carried out in [1] for the case of violin bowing.

Preliminary achievements obtained for blowing pressure estimation showed promising possibilities for indirect acquisition from microphone in real performance contexts, leading to the easier construction of multi-modal databases more suited to study higher-level playing styles. Automatic detection of fingering information [15] is a very interesting addition to future developments of the system.

7. ACKNOWLEDGMENTS

We are extremely grateful to Luthier Josep Tubau for his great help in designing the recorder, and to recorder performance Professor Joan Izquierdo who kindly collaborated during the recordings.

8. REFERENCES

- [1] E. Maestre, *Modeling instrumental gestures: An analysis/synthesis framework for violin bowing*, Ph.D. thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2009.
- [2] J. Chen, D. Laurin, J. Smith, and J. Wolfe, "Vocal tract interactions in recorder performance. 19 th," in *International Congress on acoustics Madrid*, 2007, pp. 1–6.
- [3] J. Martin, *The acoustics of the recorder*, Moeck, 1994.
- [4] NH Fletcher, "Acoustical correlates of flute performance technique," *Acoustical Society of America*, vol. 57, no. 1, 1975.
- [5] N. Montgermont, B. Fabre, and P. de la Cuadra, "Flute control parameters: fundamental techniques overview," in *International Symposium on Music Acoustics*, 2007.
- [6] C. Traube, P. Depalle, and M. Wanderley, "Indirect acquisition of instrumental gesture based on signal, physical and perceptual information," in *Proceedings of the 2003 conference on New interfaces for musical expression*. Citeseer, 2003, pp. 42–47.
- [7] B. Schoner, C. Cooper, C. Douglas, and N. Gershenfeld, "Data-driven modeling of acoustical instruments," *Journal of New Music Research*, vol. 28, no. 2, pp. 81–89, 1999.
- [8] A. Perez, *Enhancing spectral synthesis techniques with performance gestures using the violin as a case study*, Ph.D. thesis, 2009.
- [9] A. Perez and J. Bonada, "The bowed tube: a virtual violin," in *Proceedings of the 2010 conference on New interfaces for musical expression*, 2010.
- [10] F. García, L. Vincelas, J. Tubau, and E. Maestre, "Acquisition and study of blowing pressure profiles in recorder playing," in *Proceedings of the 2011 conference on New interfaces for musical expression*, 2011.
- [11] P. de la Cuadra, B. Fabre, N. Montgermont, and C. Chafe, "Analysis of flute control parameters: A comparison between a novice and an experienced flautist," *Acta Acustica united with Acustica*, vol. 94, no. 5, pp. 740–749, 2008.
- [12] NH Fletcher and L.M. Douglas, "Harmonic generation in organ pipes, recorders and flutes," *J. Acoust. Soc. Am*, vol. 68, no. 3, pp. 767–771, 1980.
- [13] X. Serra and J. Smith III, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.
- [14] R.C. Maher and J.W. Beauchamp, "Fundamental frequency estimation of musical signals using a two-way mismatch procedure," *Journal of the Acoustical Society of America*, vol. 95, no. 4, pp. 2254–2263, 1994.
- [15] V. Verfaillie, P. Depalle, and M.M. Wanderley, "Detecting overblown flute fingerings from the residual noise spectrum," *The Journal of the Acoustical Society of America*, vol. 127, pp. 534, 2010.

NONLINEAR ALLPASS LADDER FILTERS IN FAUST

Julius O. Smith and Romain Michon*

Center for Computer Research in Music and Acoustics
(CCRMA) Stanford University
Palo Alto, CA 94305, USA
jos@ccrma.stanford.edu

ABSTRACT

Passive nonlinear filters provide a rich source of evolving spectra for sound synthesis. This paper describes a nonlinear allpass filter of arbitrary order based on the normalized ladder filter. It is expressed in FAUST recursively in only two statements. Toward the synthesis of cymbals and gongs, it was used to make nonlinear waveguide meshes and feedback-delay-network reverberators.

1. INTRODUCTION

Many musical instruments have important nonlinear effects influencing their sound. In particular, cymbals and gongs exhibit evolving spectra due to nonlinear coupling among their resonant modes [1]. One effective method for efficiently synthesizing such sounds is using the digital waveguide mesh [2, 3] terminated by nonlinear allpass filters [4]. The mesh models linear wave propagation in 2D, while the nonlinear allpass provides nonlinear coupling of the modes of vibration in a way that conserves signal energy, and therefore does not affect damping (which is introduced separately via lowpass filters at selected points in the mesh). Thus, nonlinear allpass filters provide a valuable tool for nonlinear mode combination while preserving stability and keeping decay-time separately controllable.

2. PASSIVE NONLINEAR FILTERS

2.1. First-Order Switching Allpass

The passive nonlinear filter described in [4] was based on the idea of terminating a vibrating string on two different springs k_1 and k_2 , as shown in Fig. 1. The switching spring-constant creates a nonlinearity in the string-spring system. Importantly, the switching from one spring to the other only occurs when the spring displacements are zero, so that energy is not affected. (The potential energy stored in a spring k_i displaced by x_i meters is given by $k_i x_i^2 / 2$ [5].)

In an ideal vibrating string with wave impedance R , terminating the string by an ideal spring k_i provides an *allpass reflectance* at the end of the string for traveling waves [5]. That is, reflected displacement waves $y^-(t)$ at the termination are related to the incident waves $y^+(t)$ by

$$Y^-(s) = Y^+(s)H_i(s)$$

* CCRMA visiting researcher from Saint Étienne University, France, supported by the ASTREE Project

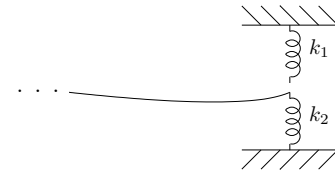


Figure 1: Vibrating string terminated by two different springs k_1 and k_2 . Only one spring is active at a time.

where $H_i(s)$ is the (Laplace-domain) transfer function of the allpass filter

$$H_i(s) = \frac{s - k_i/R}{s + k_i/R}.$$

Replacing the ideal string by a digital waveguide [5] and digitizing the spring reflectance $H_i(s)$ via the bilinear transform [5] yields the digital reflectance

$$H_i(z) = -\frac{a_i + z^{-1}}{1 + a_i z^{-1}}, \quad a_i = \frac{k_i - 2Rf_s}{k_i + 2Rf_s}$$

where f_s denotes the sampling rate in Hz. While the digital reflectance remains an allpass filter due to properties of the bilinear transform, energy conservation is only approximately obtained, except when the allpass state variable happens to be exactly zero when the coefficient a_i is switched from a_1 to a_2 or vice versa.

2.2. Delay-Line Length Modulation

Since allpass filters are fully characterized by their time-delay at each frequency, the switching allpass of the previous section can be regarded as a form of nonlinear delay-line length modulation in which the delay line switches between two allpass-interpolated lengths (different at each frequency in general).

Delay-line length modulation has been used previously to simulate nonlinear string behavior. For example, the length modulations due to tension variations have been addressed [6]. Additionally, it has long been recognized that the highly audible nonlinearity of the sitar is due to the continuous length modulation caused by its curved bridge [1]. Similarly, the tambura nonlinearly modulates its string length between two lengths via a cotton thread near the bridge [1]. In digital waveguide models such as *Sitar.cpp* in the Synthesis ToolKit (STK) [7], delay-line length is modulated without careful regard for energy conservation; this normally works out fine in practice because lengthening a delay-line is energy conserving when the new samples are zero, and shortening the delay-line is typically a bit lossy and never energy-creating.

3. NONLINEAR ALLPASSES OF ARBITRARY ORDER

We propose to extend the nonlinear switching allpass in two ways:

1. Any order allpass can be used (not just first order).
2. Any kind of coefficient modulation can be used (not just switching between two values at zero crossings of some state variable).

Our method is based on the Normalized Ladder Filter (NLF) [8]. Such filters can be derived from digital waveguide filters by using normalized traveling waves in place of ordinary physical traveling waves [5], where the normalization is chosen so that the square of the traveling-wave amplitude equals the power associated with that sample in the waveguide network.

Figure 2 shows the first-order NLF allpass as it is typically drawn [9, 5].

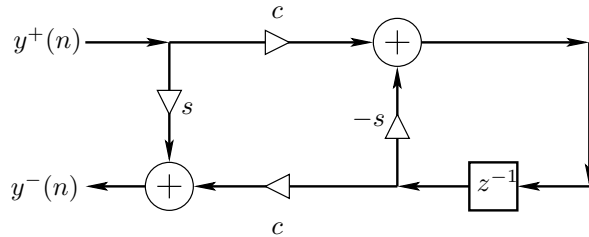


Figure 2: First-order normalized ladder allpass filter, with coefficients $c = \cos(\theta)$ and $s = \sin(\theta)$, $\theta \in [-\pi, \pi]$.

Figure 3 shows the same filter as it is depicted in the block diagram rendered by “faust -svg” for the FAUST expression

```
process =
  _ <: *(s), (*(c) : (+:_ ~ (*(s))):_, mem*c:+
```

The function `allpassnn(1)` is equivalent to this in the FAUST distribution (`filter.lib` after 2/2/11).

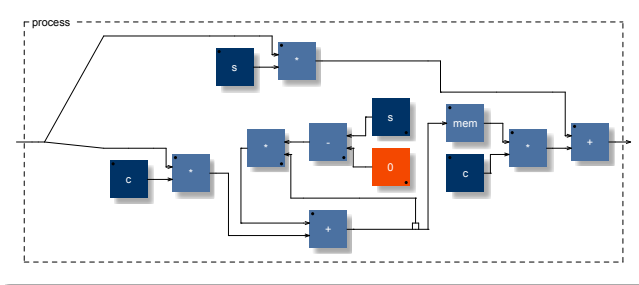


Figure 3: First-order normalized ladder allpass filter as drawn by `faust -svg`.

A general property of allpass filters is that each delay element can be replaced by an allpass to produce another allpass (a unit-circle to unit-circle conformal map of the transfer function). If the delay element in Fig. 2 is replaced by itself at the output of another first-order allpass of the same form (Fig. 2), then a second-order NLF allpass is obtained. This second-order allpass and a series delay element can then be used to replace the delay element of Fig. 2 in the same way to produce a third-order NLF allpass,

and so on. Thus, by induction, we may construct an allpass of arbitrary order as the recursive embedding of first-order allpasses of the form shown in Fig. 2. This recursive construction works also for ladder/lattice allpasses of the Kelly-Lochbaum, two-multiply, and one-multiply forms [9, 10, 5].

In FAUST, NLF allpass filters of arbitrary order are conveniently specified by means of the pattern-matching facility:

```
allpassnn(0,tv) = _;
allpassnn(n,tv) = _ <: *(s), (*(c) :
  (+ : allpassnn(n-1,tv)) ~ (*(s))
  : _, mem*c: +
  with {
    c=cos(take(n,tv)); s=sin(take(n,tv));
  };
```

This is the full definition of `allpassnn()` in `filter.lib`. Similar two-statement FAUST functions have been added to `filter.lib` for ladder/lattice allpasses of the Kelly-Lochbaum, two-multiply, and one-multiply forms.

Figure 4 shows the block diagram generated for the second-order NLF allpass specified as `allpassnn(2,tv)`.

4. APPLICATIONS

4.1. Lossless Nonlinear Spectral Expansion

The following FAUST program provides a simple illustration of the lossless spectral spreading property of nonlinear allpass filtering:

```
import("filter.lib");
N = 3; // allpass filter order
process = sineswing : nl_allpass
with {
  sineswing = 1-1':nlf2(f1,1):_,!;
  M=1024; // FM period in samples
  f1 = 0.05*SR*(1+cos(2*PI*index(M)/M));
  nl_allpass(x) = allpassnn(N,coeffs(g*x),x);
  coeffs(x) = par(i,N,x); // signal = coeff
  g = 1 : delay(M,M-1) : *(0.1*index(M)/M);
};
```

In this example, an LFO-frequency-modulated sinusoid is fed to a nonlinear allpass whose coefficients, all of which are proportional to the input signal, begin increasing according to g after one FM cycle. Figure 5 shows the spectrogram of the first two FM cycles (2048 samples). Over the first cycle, there is a pure tone cycling in frequency between 0.1 and 0 times the sampling rate. Over the next FM cycle, the nonlinearity range ramps up from 0 to 0.1, and higher-order harmonics appear due to the nonlinearity, which “brightens” the spectrum. At the same time, signal energy is unaffected, so the nonlinear allpass may be used in a nearly lossless feedback loop, as is common in waveguide string models, for example. For a proof of the energy invariance property of the nonlinear allpass, see, e.g., [9] for a mathematical derivation, or [5] for a physical formulation.

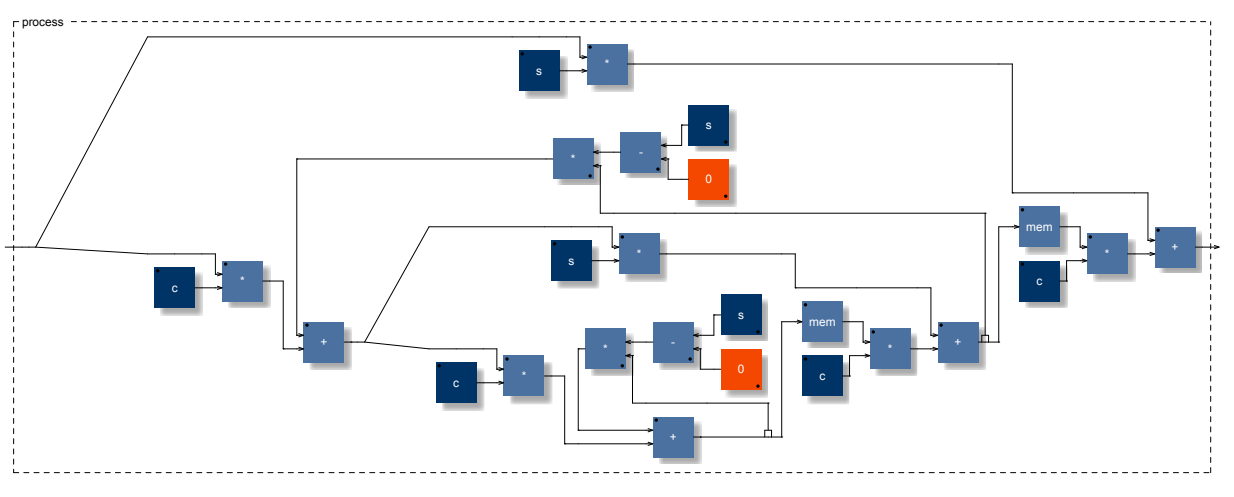


Figure 4: Second-order NLF allpass as drawn by *faust -svg*.

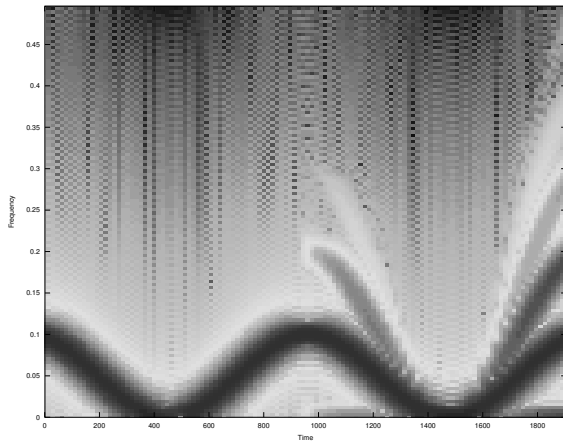


Figure 5: Spectrogram of the first 2048 samples.

4.2. Nonlinear Waveguide Mesh

A simple example terminating a square waveguide mesh with nonlinear allpass filters is shown in Figures 6 and 7, generated from the following FAUST source:

```
import("effect.lib"); // for mesh_square()
nlmesh(N,NA,x)=mesh_square(N)~(apbank(4*N,x))
with {
  coeffs(x)=par(i,NA,x); // e.g.
  apbranch(i,x) = allpassnn(NA,coeffs(x),x);
  apbank(M,x) = bus(M)
    : par(i,M-1,apbranch(i)),
      apbranch(M-1) + x;
};
N=2; // mesh order (nonnegative power of 2)
NA=1; // allpass order (any positive integer)
process = nlmesh(N,NA);
```

The input signal is added into a corner of the mesh, where all modes are excited.

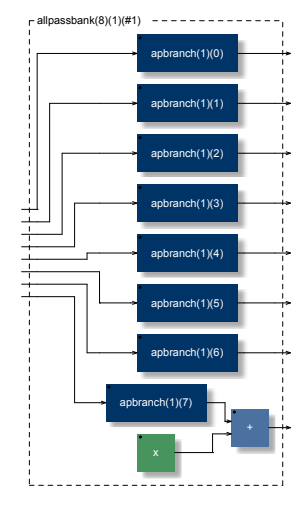


Figure 7: Nonlinear allpass bank showing mesh corner excitation. This is an expansion of the block is labeled “apbank(8)(#1)” in Fig. 6.

4.3. Nonlinear Feedback Delay Network

It was reasoned that the higher-order nonlinear allpass might enable structures simpler than a full waveguide mesh for simulating nonlinearly coupled modes. Thus, another test along these lines was to insert the nonlinear allpass into each lane of a Feedback Delay Network (FDN) reverberator (trivially modifying `fdnrev0` in `effect.lib`). The nonlinear “reverberator” so obtain was then tested by listening to its impulse response. While simple cymbals were not obtainable for the cases tried, some very nice metallic-plate synthesis was obtained, especially when using small delay-line lengths in the FDN reverb. An interesting sonic phenomenon

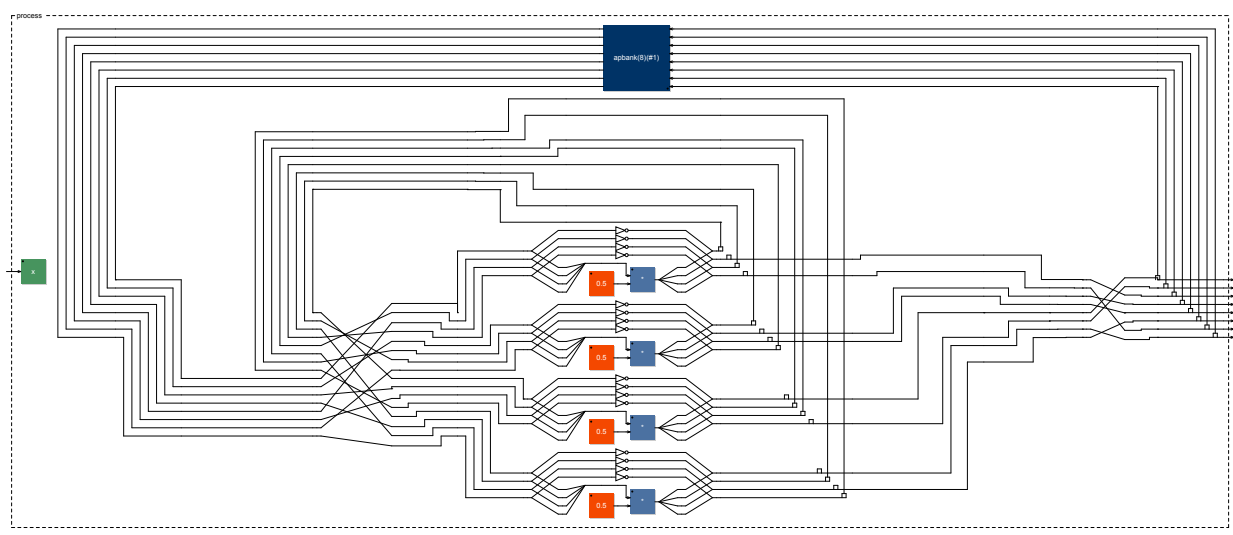


Figure 6: 2×2 square waveguide mesh terminated on nonlinear allpass filters around its rim. The topmost block is labeled “apbank(8)(#1)” and is expanded in Fig. 7. An interactive (clickable) block diagram may be generated from the above FAUST code using the utility shell-script `faust2firefox` distributed with FAUST. The input signal, labeled x on the left, is summed with the output of the eighth nonlinear allpass filter, as shown in Fig. 7. This corresponds to exciting one corner of the mesh.

obtained was a rising perceptual pitch as the impulse density was increased. In other words, a faster “drum roll” has a higher spectral centroid. A valuable performable dimension is the amount of nonlinearity, implementable by a scale factor on the reflection coefficients, such as the coefficient g in the example of §4.1. When g is zero, the allpass reduces to a linear pure delay, and increasing it from zero gradually introduces the nonlinearity.

4.4. Nonlinear Tubes and Strings

In a companion paper [11], we report on applications of the nonlinear allpass to FAUST implementations of digital waveguide and modal synthesis instruments. Especially nice results were obtained for the nonlinearly delay-modulated clarinet and harpsichord.

5. CONCLUSIONS

The nonlinear allpass based on a recursively defined normalized ladder filter was found to be useful for the nonlinear synthesis of metallic plates by terminating a waveguide mesh on its rim and by inserting it in the feedback paths of an FDN reverberator. By varying the degree of nonlinearity, a useful performance dimension is obtained.

6. REFERENCES

- [1] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*, 2nd Edition, Springer Verlag, New York, 1998.
- [2] Scott A. Van Duyne and Julius O. Smith III, “Physical modeling with the 2-D digital waveguide mesh,” in *Proc. 1993 Int. Computer Music Conf., Tokyo*, Tokyo, Japan, Sept. 1993, pp. 40–47.
- [3] Lauri Savioja and Vesa Välimäki, “Reducing the dispersion error in the digital waveguide mesh using interpolation and frequency-warping techniques,” vol. 8, no. 2, pp. 184–194, Mar. 2000.
- [4] John R. Pierce and Scott A. Van Duyne, “A passive nonlinear digital filter design which facilitates physics-based sound synthesis of highly nonlinear musical instruments,” *J. Acoustical Soc. of America*, vol. 101, no. 2, pp. 1120–1126, Feb. 1997.
- [5] Julius O. Smith III, *Physical Audio Signal Processing*, <https://ccrma.stanford.edu/~jos/pasp/>, Dec. 2010, online book.
- [6] Tero Tolonen, Vesa Välimäki, and Matti Karjalainen, “Modeling of tension modulation nonlinearity in plucked strings,” vol. SAP-8, pp. 300–310, May 2000.
- [7] Perry Cook and Gary Scavone, *Synthesis ToolKit in C++*, <http://ccrma.stanford.edu/software/stk/>, 2010.
- [8] A. H. Gray and J. D. Markel, “A normalized digital filter structure,” *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-23, no. 3, pp. 268–277, June 1975.
- [9] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*, Springer Verlag, New York, 1976.
- [10] Dana Massie, “An engineering study of the four-multiply normalized ladder filter,” vol. 41, no. 7/8, pp. 564–582, 1993.
- [11] Romain Michon and Julius O. Smith, “FAUST-STK: A set of linear and nonlinear physical models for the FAUST programming language,” in *Proc. 14th Int. Conf. on Digital Audio Effects (DAFx-11)*, Paris, France, September 19–23, 2011.
- [12] Matti Karjalainen, Juhan Backman, and J. Pölkki, “Analysis, modeling, and real-time sound synthesis of the kantele, a traditional Finnish string instrument,” New York, 1993, pp. 229–232, IEEE Press.

A CSOUND OPCODE FOR A TRIODE STAGE OF A VACUUM TUBE AMPLIFIER

Marco Fink, Rudolf Rabenstein

Chair of Multimedia Communications and Signal Processing
University Erlangen-Nuremberg
Erlangen, Germany
rabe@LNT.de

ABSTRACT

The Csound audio programming language adheres to the input-output paradigm and provides a large number of specialized commands (called opcodes) for processing output signals from input signals. Therefore it is not directly suitable for component modeling of analog circuitry. This contribution describes an attempt to virtual analog modeling and presents a Csound opcode for a triode stage of a vacuum tube amplifier. Externally it communicates with other opcodes via input and output signals at the sample rate. Internally it uses an established wave digital filter model of a standard triode. The opcode is available as library module.

1. INTRODUCTION

Like many other musical instruments, electric guitars require an additional amplifier. However, it is understood among guitarists that increasing the sound volume is only one of its several purposes. Others are spectral shaping by filter networks and the creation of nonlinear distortion by overdriving the amplifier elements. For historical and technical reasons, tube amplifiers are still popular despite of some obvious practical disadvantages over their solid state counterparts. Quite a few music genres with dominant electric guitars have evolved during the last fifty years and most of them are inextricably linked to certain tube amplifier models. Although many of these are still produced, vintage objects are highly priced collectibles.

As stand-alone electro-acoustic devices, classical vacuum tube guitar amplifiers are incompatible with the digital world of today's music production. In the original tube amplifier models, there is no MIDI control nor are there presets or software updates. Other obvious disadvantages concern weight, size, and waste heat. To preserve the original sound of tube amplifiers and still benefit from the advantages of digital technology, numerous approaches have been developed for imitating the behavior of electrical circuits with digital signal processing methods. Conventional programs for circuit simulation are of limited avail, since the challenge in amplifier modeling is at least threefold: first the exact replication of nonlinear effects, second the real-time performance, and third the minimization of the latency between sound input and output.

The term virtual analog (VA) has been coined for the substitution of analog circuits by real-time digital hard- and software. A recent description on virtual analog models in general is given in [1] while digital techniques for modeling vacuum tube amplifiers are reviewed in [2]. Both articles contain vast references to earlier literature.

Among the various approaches discussed in [2] are circuit simulation-based techniques which start from the schematic circuit diagram. One way to proceed is to apply methods from circuit

analysis and to derive a set of coupled nonlinear differential equations. The skillful implementation of robust numerical solvers leads to real-time algorithms for the faithful reproduction of amplifier sounds. Details can be found e.g. in [3–5] and references therein.

Another approach for converting a circuit diagram with possibly nonlinear elements into a digital signal processing algorithm is the classical wave digital filtering method [6]. Applications to musical signal processing are described in [7, 8] for physical modeling of musical instruments and in [9] for virtual analog modeling. Wave digital models for tube amplifiers have been presented in [10, 11] for a triode stage and in [12, 13] for the subsequent transformer and loudspeaker. The model from [11] is used here for the implementation of a Csound opcode for a triode stage.

Csound is an audio programming language which provides a multitude of commands (so-called opcodes) for all kinds of audio signal generation and processing [14]. In simple terms, the opcodes accept input signals and parameters and provide output signals. Circuit modeling, on the other hand, has to consider connections of ports, rather than inputs and outputs (see [15] for a detailed discussion). Thus Csound is not directly suitable for modeling the mutual dependencies between voltages and currents in electrical circuits. There are – to the knowledge of the authors – no models for virtual analog audio effects at the component level in Csound.

Sec. 2 briefly reviews the triode model from [11]. Basic concepts of wave digital filters are recalled in Sec. 3, while Sec. 4 introduces the Csound audio programming language. Sec. 5 gives a short review of different variable types and Sec. 6 discusses some issues of real-time implementation of linear and non-linear wave digital filters. The Csound opcode for the triode stage is presented in detail in Sec. 7 along with an example.

2. TRIODE STAGE

The implemented triode stage is a classical grounded cathode amplifier as shown in Fig. 1. Since it is a standard triode amplifying stage, no details are presented here. They can be found in classical references e.g. [16–18], or in more recent presentations of vacuum tube circuits such as [19] or a monograph on SPICE models [20].

Digital models of this triode stage have been considered in [5] by analysis of the corresponding nonlinear circuit and in [10, 11] with wave digital filters. The latter approach is adopted here due to its versatility for building modular structures.

3. WAVE DIGITAL FILTERS

A standard method to convert continuous-time transfer functions into discrete-time transfer functions is the bilinear transformation.

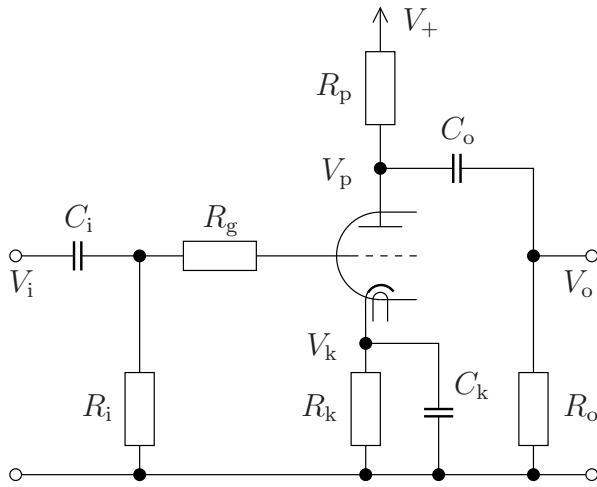


Figure 1: Standard triode stage (adapted from [10]). It is usually followed by a gain stage consisting of another triode in the same envelope.

It is easily derived as the numerical solution of the integrals in a continuous-time state space structure with the trapezoidal rule. It could also be used in component modeling for the integrations involved in the voltage-current law at capacitances and vice versa the current-voltage law at inductances. However, connecting such discrete-time models leads to delay-free loops which renders the resulting discrete-time algorithm as non-computable.

An escape to this situation has been noticed long ago and lead to a method for designing digital filters (wave digital filters) from analog counterparts in the early 1970s. A unifying treatment of theory and application of wave digital filters is given in a classical paper [6]. Modern descriptions of the wave digital principle as a tool for numerical integration and modeling are given in [7,21–23]. The use of wave variables for block-based physical modeling is discussed in [8]. Different wave digital implementations of the triode model from Fig. 1 have been presented in [10, 11]. The model from [11] with a purely resistive load R_o provides the basis for the implementation as a Csound opcode (see Fig. 2).

The fundamental difference between the wave digital model according to Fig. 2 and the circuit diagram from Fig. 1 is the nature of the variables which connect the individual blocks in Fig. 2. The variables for the description of circuit diagrams like Fig. 1 are voltages and currents, also called Kirchhoff variables. In general, they cannot be classified into input and output signals, since a variation of a voltage implies also a variation of the corresponding current and vice versa.

Wave variables are computed from and can be converted into Kirchhoff variables (see [7,8,22,23] for details). However, since they come in pairs of an incident and a reflected wave, they induce a natural processing order. A block diagram of the wave digital implementation from [11] is shown in Fig. 2. The ports with the two-way arrows indicate wave variables, V_i and V_o are input and output voltages.

4. CSOUND

Csound is an audio programming language which supports the programmer in such intricate audio issues as sampling rate synchro-

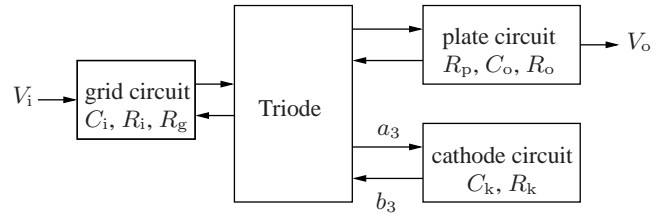


Figure 2: Block diagram of the wave digital model from [11]. The designation of the resistors and capacitors is the same as in Fig. 1. The wave variables a_3 and b_3 correspond to Fig. 4.

nization, timing control, and buffer management. It is described in [14] and in various web resources accessible from [24].

Csound is input-output oriented in the sense that existing high level commands (opcodes) read samples from an input signal and write it to an output signal. An excerpt from a generic Csound program looks like

```
asig2 opcode1 asig1 parameters
asig3 opcode2 asig2 parameters
```

The command `opcode1` processes the input signal `asig1` and produces the signal `asig2` which then serves as input signal for `opcode2`. These opcodes are processed in a program loop which is executed at the audio sample rate (see e.g. [14]).

The order of the commands determines the signal flow. It is shown as a block diagram in Fig. 3 which clearly demonstrates the input-output character of the Csound processing paradigm. In contrast to Kirchhoff variables, an output signal is not affected when it is connected to another opcode as an input signal.

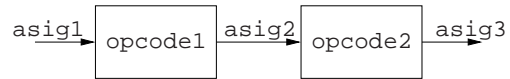


Figure 3: Basic Csound signal flow.

5. KIRCHHOFF VARIABLES, WAVE VARIABLES, AND SIGNALS

Sections 2, 3, and 4 of this article each use a different type of variables, voltage and current in Sec. 2, wave variables in Sec. 3, and in Sec. 4 input and output signals represented by the audio samples in `asig1...asig3`. The Kirchhoff variables voltage and current can be converted into incident and reflected wave variables and vice versa. However, the communication of these types of variables with the input-output signals in Csound requires some consideration. The relations between these different types of variables are reviewed in [8], a recent and very detailed account is found in [15].

In spite of the different types of variables used in circuit theory, wave digital filters, and systems theory it is possible to implement a wave digital model of the triode stage as a Csound opcode. The reason lies in the design of the analog circuit itself.

The triodes stage is designed for high input impedance. Therefore the voltage V_i can be regarded as an input signal which is not affected by feeding it to the tube model. On the output side, the triode stage may be connected to another amplifier stage or a tone stack. Depending on the input impedance of the subsequent

load, the plate current may be affected by this connection. This situation is modeled by including a load resistor R_o into the plate circuit. The voltage V_o then serves as the corresponding output signal. Some implementation issues resulting from this mixed-variable approach are discussed in Sec. 6.

The assumption of a high input impedance and a resistive load are reasonable simplifications. More general cases can be modeled by an input voltage source with finite internal impedance and a reactive (complex-valued) output impedance.

6. IMPLEMENTATION ISSUES

The implementation of the concepts discussed above in a procedural language like C requires to convert the block diagram from Fig. 2 into a sequence of computational instructions. Most of these steps are described in the standard literature on wave digital filters (e.g. [6]) and are mentioned here only very briefly for the linear case. Then the nonlinear properties of the triode are discussed.

6.1. Linear Elements

The conversion of electrical circuits with linear elements into a wave digital structure is shown here for the example of the cathode circuit in Figs. 1 and 2. The parallel arrangement of a resistance and a capacitance requires a so-called parallel adaptor for the correct calculation of the wave variables for the communication with the triode circuit.

The structure of the required constrained three-port parallel adaptor and the equations for the calculation of the adaptor coefficients are given in [6, Table 6]. The reflection free port is the one connected to the triode circuit. It is called port 3 and has the wave variables a_3 and b_3 . Port 1 with the wave variables a_1 and b_1 is used for the connection of the capacitance model for C_k which is a simple delay element (see [6, Table 1]). Port 2 is used for the resistance R_k . Its implementation is particularly easy, since a resistance does not reflect any waves. Therefore the incident wave a_2 (w.r.t. port 2) is zero and the reflected wave b_2 does not need to be computed. These considerations lead to the simplified structure of the cathode circuit from Fig. 4.

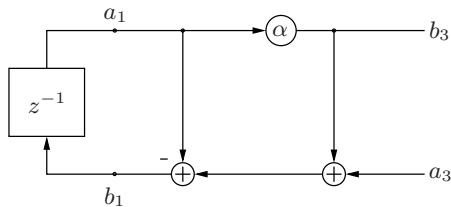


Figure 4: Signal flow diagram of the cathode circuit. The wave variables a_3 and b_3 correspond to Fig. 2.

Although the resistance R_k does not reflect any waves, its effect and numerical value are nevertheless considered in the calculation of the adaptor coefficient α . It depends also on the capacity C_k and on the sampling instant T

$$\alpha = \frac{R_3}{R_1} = \frac{2}{T} R_k C_k . \quad (1)$$

Exemplarily the software implementation of the cathode circuit according to Fig. 4 is shown below in C-style notation.

```
//Calculate reflected wave
a_1 = state_Ck;
b_3 = alpha*a_1;
//Calculate effects of incident wave
b_1 = a_3 + b_3 - a_1;
state_Ck = b_1;
```

The grid circuit and the plate circuit from Fig. 2 are implemented along the same lines.

6.2. Nonlinear Elements

The characteristic sound of a tube amplifier is caused by the nonlinear behavior of the triode. Two effects are important, the grid-to-cathode diode characteristic and the nonlinearity of the plate current as a function of the plate voltage and the grid voltage. Both effects are realized separately in the implementation of the block *Triode* from Fig. 2. The connection between these nonlinear processing elements and the wave variables at the respective ports is shown in the detailed triode model in Fig. 5.

6.2.1. Grid-to-Cathode Diode

The grid-to-cathode diode characteristic can be modeled as a nonlinear resistor R_d . Two different models have been suggested in [11] and both are implemented here. The first one simply switches between two constant resistor values, i.e.

$$R_d = \begin{cases} R_{low} & \text{if } V_{gk} > 0 \\ R_{high} & \text{else} \end{cases} . \quad (2)$$

The second model is an implementation of the Child-Langmuir law. The parameters for these models (e.g. R_{low} and R_{high}) can be set by the user in a configuration file. Both diode models are controlled by the grid-to-cathode voltage V_{gk} which is obtained as the mean of the incoming and reflected waves at the adaptor port to which the diode is connected.

6.2.2. Plate Current

For the nonlinear plate current, Koren's model has already been used successfully in [5, 10]. It expresses the plate current $I_{pk} = f(V_{gk}, V_{pk})$ as a nonlinear function of the grid-to-cathode voltage V_{gk} and the plate-to-cathode voltage V_{pk} . Substituting this nonlinear function for I_{pk} in the corresponding wave variables

$$a_{pk} = V_{pk} + R_0 I_{pk}, \quad (3)$$

$$b_{pk} = V_{pk} - R_0 I_{pk}, \quad (4)$$

gives an implicit relation for the unknown plate-to-cathode voltage V_{pk} and an explicit relation for the reflected wave b_{pk} [10]

$$0 = V_{pk} + R_0 f(V_{gk}, V_{pk}) - a_{pk}, \quad (5)$$

$$b_{pk} = V_{pk} - R_0 f(V_{gk}, V_{pk}). \quad (6)$$

The implicit relation (5) is solved from an initial guess (e.g. the previous value) by minimizing the error function

$$e = V_{pk} + R_0 f(V_{gk}, V_{pk}) - a_{pk} \quad (7)$$

in an iterative procedure as suggested in [10]. The secant method has been used in Fig. 5 with the iteration index m . For values

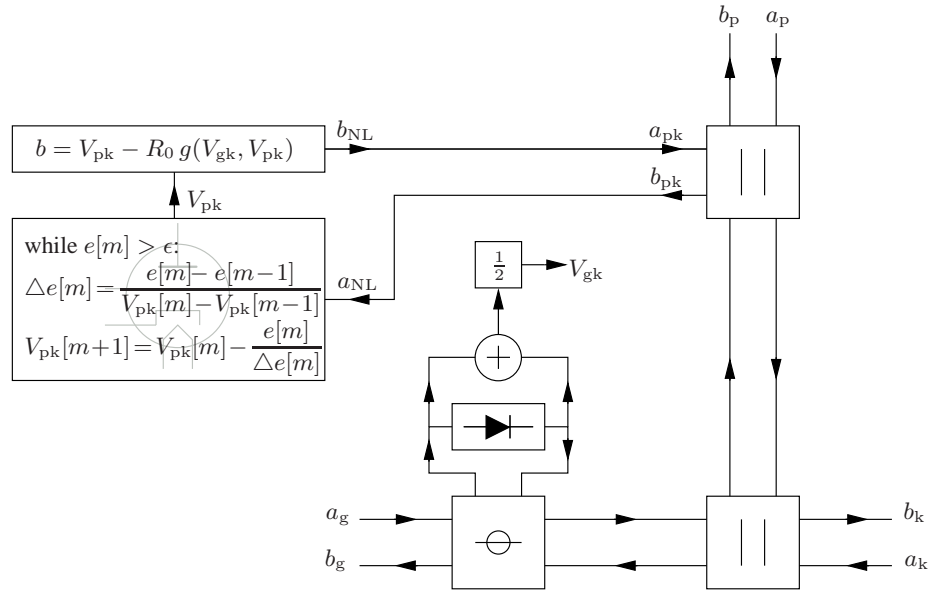


Figure 5: Nonlinear triode model. The circuit shown here implements the block *Triode* from Fig. 2. The ports with the wave variable pairs (a_g, b_g) , (a_k, b_k) , and (a_p, b_p) connect to the grid, cathode, and plate circuits, respectively. The grid-to-cathode diode characteristic discussed in Sec. 6.2.1 is realized by the block connected to the series adaptor and labeled with a diode symbol. The plate current results from the iterative solution of Koren’s model for V_{pk} in the block connected to the upper parallel adaptor, see Sec. 6.2.2.

within the usual operation range, these nonlinear relations are pre-calculated and stored in a look-up table.

Nonlinear mappings of audio signals may produce spectral components above the human hearing range. In discrete-time processing, aliasing can fold these components back onto audible frequencies unless oversampling is applied. Also this implementation of the triode model employs internal oversampling for the evaluation of the nonlinear tube model. The locations for up-sampling and down-sampling are straight behind the signal input V_i and before the signal output V_o (see Fig. 2). If required, it is possible to move the sampling rate conversion closer to the nonlinearities.

7. CSOUND OP CODE

To turn existing C-code into opcodes for Csound, the communication of the C-code with the Csound programming environment has to be established. For this purpose, Csound provides a fixed interface [14, chapters 31 and 32]. Using this interface, the wave digital model of the triode stage described above has been implemented as a C-program for real-time operation. The result is a shared library which has so far been built and tested for UNIX environments. It supplies a new Csound opcode called `tube` for the simulation of a vacuum tube triode stage.

The `tube` opcode is ready to use and implements the triode model with standard values for its parameters (e.g. component values, load resistance, supply voltage, oversampling rate, parameters of the tube nonlinearities, etc.). To provide maximum flexibility for the user, these standard values can also be overridden by the content of an external configuration file. The Csound opcode `tube` is available for download at [25]. A possible use case for the `tube` opcode is shown below.

```
kamp = 0.8;
icps = 440;
ifn = 0;
imeth = 1;
apluck pluck kamp, icps, icps, ifn, imeth;
atube tube apluck, sr, "config.txt";
alp butterlp atube, 5000;
ares butterbr alp, 1200, 50;
out ares;
```

A dry input signal for the tube stage is generated by the Csound opcode `pluck` which is controlled via the parameters `kamp` for the amplitude, `icps` for the oscillation frequency, `ifn` to choose an initial buffer content and `imeth` to control the decay (see [24]). `pluck` implements the Karplus-Strong algorithm and generates the signal `apluck`.

The sample of a string-like sound `apluck` and the user defined sample rate `sr` are fed to the opcode `tube`, creating the `atube` output signal. Optionally the filename of an external configuration file can be handed to the opcode to let the user set the modeling parameters.

A simple loudspeaker emulation, proposed by [10] is applied via other standard opcodes, realizing a lowpass filter at 5 kHz and a notch filter with a bandwidth of 50 Hz at a center frequency of 1200 Hz. At last the resulting sound sample is written to the variable `out`, defining Csound’s realtime or filebased output.

8. RESULTS

Although the presented implementation of the triode model can only judged by listening, a few technical results are reported here. They concern the nonlinear model for the grid-to-plate current and its effects on a sinusoidal input signal, the amount of aliasing for

different sampling rates, and finally a pointer to some listening examples is given.

8.1. Nonlinear Model for the Grid-to-Plate Current

The basic effect leading to the typical tube behaviour is the above mentioned nonlinear current characteristic. The implementation of the nonlinear model from Sec. 6.2.2 has been evaluated by measuring the V_{pk} - I_{pk} curves at the port (a_{pk} , b_{pk}) from Figure 5. The results are plotted in Figure 6 together with the theoretical curves predicted by Koren's formulas. Visually these curves cannot be distinguished. An analytical analysis of the absolute difference showed a maximum deviation of about $3 \cdot 10^{-6}$ mA within the usual operation range. ($V_{pk} \in [0V, \dots, 300V]$ and $V_{gk} \in [-3V, \dots, 0V]$)

The resulting spectral effect of the nonlinear behaviour is the appearance of additional harmonics. As a demonstration, a sinusoidal input signal with a frequency of 440 Hz and a slowly rising amplitude was generated.

The amplitude increased linearly from zero to full scale in seven seconds, where full scale corresponds to an amplitude of 3 V of the input signal V_i in Figure 2. This input signal was processed with the opcode and then plotted as a spectrogram (44.1 kHz sample rate, FFT length 2048).

Figure 7 shows on the top the effect of additional spectral components, explaining the rich sound of vacuum tube amplifiers. The envelope of the distorted output signal is plotted on the bottom. The asymmetrical shape of the envelope results from the tube-typical one-sided clipping whenever the amplitude of the input signal V_i exceeds the negative bias of V_{gk} implemented by the cathode resistance R_k in Figure 1.

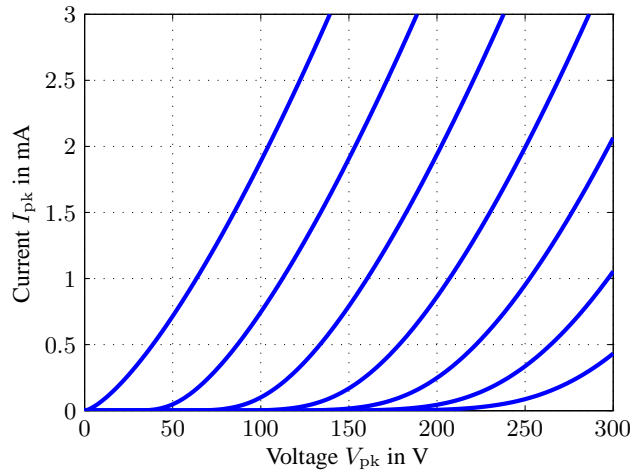


Figure 6: V_{pk} - I_{pk} characteristic curve, measured in the nonlinear part of the WDF circuit from Fig. 5. The curves are obtained for V_{gk} from $-3V$ to $0V$ in steps of $0.5V$.

8.2. Oversampling

Figure 8 shows the spectrum of a distorted 5 kHz sine with an amplitude of 2.5 V, sampled at 44.1 kHz. The upper graph shows a dominant peak at the fundamental frequency and further peaks at the positions of the first three harmonics. But also other peaks

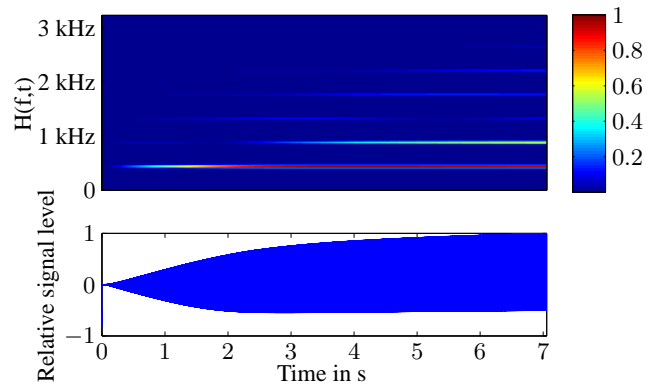


Figure 7: Spectrogram of the response to a 440 Hz sinusoidal input signal with slowly increasing amplitude. Top: Spectrogram with magnitude normalized to unity. Bottom: Envelope of the output signal.

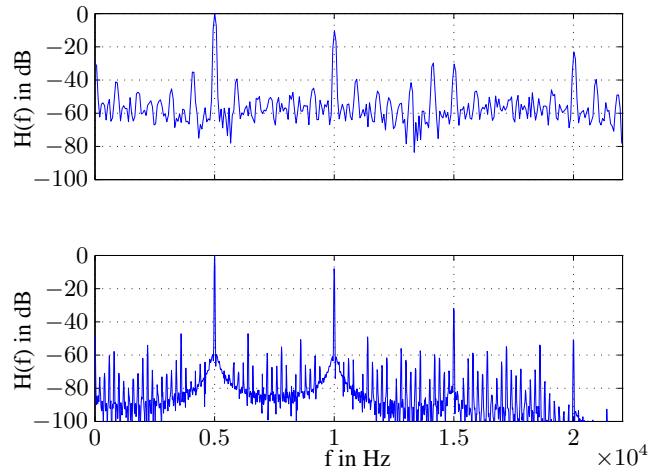


Figure 8: Spectrum of the response to a 5kHz sinusoidal input signal with an amplitude of 2.5V at a sample rate of 44.1kHz without and with four times oversampling.

are visible in the range of -30 to -50 dB. They result from higher harmonics which are subject to aliasing. This effect of unwanted spectral components is less distinctive in the graph below where four times oversampling was used.

8.3. Listening Examples

To meet the actual use of the opcode, simulating the sound of a tube amplifier, also an acoustical analysis was done. Four sound clips, trying to demonstrate the different genres shown in Table 1, were recorded with a ESP LTD F-50 guitar with a common intern Realtek HD sound card at 44.1 kHz. These sound clips were processed with the opcode with the example configuration file. Only the preamplifier gain value was edited to achieve different distortion levels, fitting the musical genre. The virtual preamplifier maps the guitar signal to the input signal V_i shown in Figure 2.

Genre	gain
Blues	6
Rockabilly	8
Rock	10
Metal	15

Table 1: Available sound examples for the Csound plugin *tube*. A different preamplifier gain was used for each genre to achieve typical distortion levels.

9. CONCLUSIONS

The Csound opcode *tube* provides a real-time model of a triode stage in a vacuum amplifier. It is based on an existing wave digital model well documented in the literature. Special emphasis has been placed on the internal structure of the nonlinear triode circuit, where variables of different nature (voltages and currents vs. wave variables) are combined. The opcode *tube* is available for download and ready to use with standard values of the model parameters. Additionally the model can be modified by editing an external configuration file.

Acknowledgments

Many thanks to Jyri Pakarinen for pointing out relevant literature and to the anonymous reviewers for their helpful comments.

10. REFERENCES

- [1] Jyri Pakarinen, Vesa Välimäki, Federico Fontana, Victor Lazzarini, and Jonathan S. Abel, “Recent advances in real-time musical effects, synthesis, and virtual analog models,” *EURASIP Journal on Advances in Signal Processing*, 2011, Article ID 940784.
- [2] Jyri Pakarinen and David T. Yeh, “A review of digital techniques for modeling vacuum-tube guitar amplifiers,” *Computer Music Journal*, vol. 33, no. 2, pp. 85–100, Summer 2009.
- [3] Jaromir Macak and Jiri Schimmel, “Real-time guitar tube amplifier simulation using an approximation of differential equations,” in *Proc. Int. Conference on Digital Audio Effects (DAFx)*, Graz, Austria, 2010.
- [4] Jaromir Macak and Jiri Schimmel, “Real-time guitar preamp simulation using modified blockwise method and approximations,” *EURASIP Journal on Advances in Signal Processing*, 2011, Article ID 629309.
- [5] Francesco Santagata, Augusto Sarti, and Stefano Tubaro, “Non-linear digital implementation of a parametric analog tube ground cathode amplifier,” in *Proc. Int. Conference on Digital Audio Effects (DAFx)*, Bordeaux, France, 2007.
- [6] Alfred Fettweis, “Wave digital filters: Theory and practice,” *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.
- [7] Vesa Välimäki, Jyri Pakarinen, Cumhuri Erkut, and Matti Karjalainen, “Discrete-time modelling of musical instruments,” *Reports on Progress in Physics*, vol. 69, no. 1, pp. 1–78, January 2006.
- [8] Rudolf Rabenstein, Stefan Petrausch, Augusto Sarti, Giovanni De Sanctis, Cumhuri Erkut, and Matti Karjalainen, “Blocked-based physical modeling for digital sound synthesis,” *Signal Processing Magazine*, vol. 24, no. 2, pp. 42–54, March 2007.
- [9] Giovanni De Sanctis and Augusto Sarti, “Virtual analog modeling in the wave-digital domain,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 715–727, May 2010.
- [10] Matti Karjalainen and Jyri Pakarinen, “Wave digital simulation of a vacuum-tube amplifier,” in *Acoustics, Speech and Signal Processing (ICASSP), Proc. IEEE International Conference on*, May 2006, vol. 5, pp. 153–156, <http://www.acoustics.hut.fi/publications/papers/icassp-wdftube/>, accessed March 29, 2011.
- [11] Jyri Pakarinen and Matti Karjalainen, “Enhanced wave digital triode model for real-time tube amplifier emulation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 738–746, May 2010.
- [12] Jyri Pakarinen, Miikka Tikander, and Matti Karjalainen, “Wave digital modeling of the output chain of a vacuum-tube amplifier,” in *Proc. Int. Conference on Digital Audio Effects (DAFx)*, Como, Italy, 2009, pp. 55–59.
- [13] Rafael Cauduro Dias de Paiva, Jyri Pakarinen, Vesa Välimäki, and Miikka Tikander, “Real-time audio transformer emulation for virtual tube amplifiers,” *EURASIP Journal on Advances in Signal Processing*, 2011, Article ID 347645.
- [14] Richard Boulanger, Ed., *The Csound Book*, The MIT Press, Cambridge, MA, USA, 2000.
- [15] Jan C. Willems, “Terminals and ports,” *Circuits and Systems Magazine, IEEE*, vol. 10, no. 4, pp. 8–26, 2010.
- [16] Heinrich Barkhausen, *Lehrbuch der Elektronenröhren, 1. Band Allgemeine Grundlagen*, S. Hirzel Verlag, Leipzig, Germany, 9. edition, 1960.
- [17] Heinrich Barkhausen, *Lehrbuch der Elektronenröhren, 2. Band Verstärker*, S. Hirzel Verlag, Leipzig, Germany, 7. edition, 1959.
- [18] Austin V. Eastman, *Fundamentals of Vacuum Tubes*, McGraw-Hill, New York, Toronto, London, 3. edition, 1949.
- [19] Peter Dieleemann, *Theorie und Praxis des Röhrenverstärkers*, Elektor, Aachen, Germany, 2005.
- [20] Alexander Potchinkov, *Simulation von Röhrenverstärkern mit SPICE*, Vieweg+Teubner, Wiesbaden, Germany, 2009.
- [21] Karlheinz Ochs, “Passive integration methods: Fundamental theory,” *Int. J. Electron. Commun. (AEÜ)*, vol. 55, no. 3, pp. 153–163, 2001.
- [22] Stefan Bilbao, *Wave and Scattering Methods for Numerical Simulation*, John Wiley & Sons, Ltd, Chichester, England, 2004.
- [23] Julius O. Smith, *Physical Audio Signal Processing*, <http://ccrma.stanford.edu/~jos/pasp/>, accessed March 29, 2011, online book.
- [24] “Csound home page,” <http://www.csounds.com/>, accessed March 29, 2011.
- [25] “Csound opcode tube,” <http://www.lms.lnt.de/>.

GENERALIZED REASSIGNMENT WITH AN ADAPTIVE POLYNOMIAL-PHASE FOURIER KERNEL FOR THE ESTIMATION OF NON-STATIONARY SINUSOIDAL PARAMETERS

Sašo Muševič, *

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
saso.musevic@upf.edu

Jordi Bonada

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
jordi.bonada@upf.edu

ABSTRACT

This paper describes an improvement of the *generalized reassignment* method for estimating the parameters of a modulated real sinusoid. The main disadvantage of this method is decreased accuracy for high log-amplitude and/or frequency changes. One of the reasons for such accuracy deterioration stems from the use of the Fourier transform. Fourier transform belongs to a more general family of *integral transforms* and can be defined as an integral transform using a *Fourier kernel function* - a stationary complex sinusoid. A correlation between the Fourier kernel function and a non-stationary sinusoid decreases as the modulation of the sinusoid increases, ultimately causing the parameter estimation deterioration. In this paper, the generalized reassignment is reformulated for use with an arbitrary kernel. Specifically, an adaptive *polynomial-phase Fourier kernel* is proposed. It is shown that such an algorithm needs the parameter estimates from the original generalized reassignment method and that it improves the Signal-to-Residual ratio (SRR) in the non-noisy cases. The drawbacks concerning the initial conditions and ways of avoiding a close-to-singular system of linear equations are discussed.

1. INTRODUCTION

The extraction of sinusoidal parameters has been the focus of the signal processing research community for a very long time. The reasons for that are numerous: analysis for re-synthesis [1], voice analysis [2][3][4], music transcription [5], audio coding [6] and many more.

The classic model for modeling sinusoids implies a static amplitude and frequency within the time of observation [1]. Many refinements of this stationary model were developed [7][8][9] yet the fact that the bandwidth of a modulated sinusoid tends to raise proportionally with the amount of modulation imposed [10][11][12] rendered a need for estimation of the non-stationary parameters of sinusoids crucial [13]. Numerous Fourier transform based methods have emerged [14][15][16][17][18][19]. It has been shown in [19][20], that the generalized reassignment exhibits superior accuracy in the linear log-amplitude/linear frequency modulation context compared to QIFFT [15] and the generalized derivative method [16] in the linear log-AM/FM case. An additional advantage of the generalized reassignment is the ability to estimate the modulation parameters of arbitrary order, whereas others (except

method using distribution derivatives [18]) were designed to work only in the linear log-AM/FM context.

The generalized reassignment [19] algorithm uses values of the Short-Time-Fourier-Transform (STFT) of the signal and its time derivatives (up to M-th degree) in order to produce a linear system of M complex equations. STFT is evaluated at 1 frequency only, a natural choice for which is the maximum peak frequency. Solving this system allows the estimation of M complex parameters that uniquely define the parameters of the sinusoid. A similar algorithm described in [18] only considers 1st degree time derivative of the signal and acquires the rest of the equations by considering the values of STFTs at the spectrum peak and the nearby frequency bins. A comparison of the two in identical test conditions has not yet been conducted.

In section 2, the general framework of this paper is outlined. Section 3 states the generalized reassignment [19] method in the notation adopted by [18] and removes the restriction of the static kernel, while section 4 introduces the polynomial-phase Fourier kernel in the context of the generalized reassignment. In section 5 the results of the tests identical to those in [19] are reported, while 6 rounds up the comparison of the method proposed with the generalized reassignment and proposes further work on the topic.

2. GENERAL CONSIDERATIONS

For the purpose of this paper a complex non-stationary sinusoid is defined identically as in [19]:

$$s(t) = e^{R(t)}, R(t) = \sum_{m=0}^{M-1} r_m h_m(t), \quad (1)$$

where $R(t)$ is a complex function, a linear combination of M real functions $h_m(t)$, weighted with complex parameters r_m . The real and imaginary parts of r_m are denoted by p_m, q_m respectively, yielding: $r_m = p_m + jq_m$. A natural choice for functions h_m are monomials: $h_m(t) = t^m$. In such setting, p_0 corresponds to the stationary log-amplitude and p_1 to the linear log-amplitude modulation (or first order log-amplitude modulation), while $p_i, i > 1$ corresponds to the i -th order log-amplitude modulation. Analogously, q_0 corresponds to the stationary phase, q_1 to the stationary frequency and parameters $q_i, i > 1$ to the $(i - 1)$ -th degree frequency modulation.

The Fourier transform at a particular frequency can be conveniently represented as a *dot* product of the signal under investigation with

* Research was funded by 'Slovene human resources development and scholarship fund' ('Javni sklad Republike Slovenije za razvoj kadrov in stipendije')

the function $e^{j\omega}$:

$$T_{e^{j\omega_0 t}} s(t) = F\{s(t), w_0\} = \int_{-\infty}^{\infty} s(t) e^{-j\omega_0 t} dt = \langle s, e^{j\omega_0 t} \rangle \quad (2)$$

Swapping the Fourier kernel function with an arbitrary kernel Ψ yields:

$$T_{\Psi} s = \langle s, \Psi \rangle \quad (3)$$

By choosing the kernel function to be completely arbitrary, the orthogonality of 2 random kernels and unit energy properties are lost. However, such properties are not required by the algorithm, so its use is not restricted. An appropriate selection of the set of the kernel functions is a very different matter and depends on the family of the signals under study.

3. GENERALIZED REASSIGNMENT USING A GENERIC KERNEL

The main concept of the generalized reassignment method is based on the fact the n -th degree time derivative of the signal can be represented in the following way:

$$s^{(n)}(t) = (R'(t)s(t))^{(n-1)} \quad (4)$$

In practice a window function $w(t)$ is used in order to time limit and smooth the frame under investigation.

Independently, using the integration *per partes*, Leibniz integration rule and the restriction $w(-\frac{T}{2}) = w(\frac{T}{2}) = 0$ (required by the generalized reassignment), the following useful equality can be produced (for complete derivation see [18][19][20]):

$$\frac{\partial}{\partial t} \langle s, w\Psi \rangle = -(\langle s, w\Psi' \rangle + \langle s, w'\Psi \rangle) \quad (5)$$

The complementing equality can be deduced from 4 by applying a dot product with the kernel on both sides of the first time derivative:

$$\begin{aligned} \frac{\partial}{\partial t} \langle s, w\Psi \rangle &= \langle \frac{\partial}{\partial t} s, w\Psi \rangle = \\ &= \langle R's, w\Psi \rangle = \sum_{m=1}^M r_m \langle h'_m s, w\Psi \rangle \Rightarrow \\ \sum_{m=1}^M r_m \langle h'_m s, w\Psi \rangle &= -(\langle s, w\Psi' \rangle + \langle s, w'\Psi \rangle). \end{aligned} \quad (6)$$

To compute $M - 1$ non-stationary parameters, another $M - 2$ time derivatives are required. Its computation can efficiently be performed by the following *pyramid-like* scheme:

$$\begin{aligned} &\langle sh, \Psi_G w \rangle \\ &\swarrow \quad \searrow \\ &-\langle sh, \Psi'_G w \rangle \quad -\langle sh, \Psi_G w' \rangle \\ &\swarrow \quad \searrow \quad \swarrow \quad \searrow \\ &\langle sh, \Psi''_G w \rangle + 2\langle sh, \Psi'_G w' \rangle + \langle sh, \Psi_G w'' \rangle, \\ &\vdots \end{aligned} \quad (8)$$

where $h(t)$ stands either for $h(t) = 1$ to calculate right hand side or $h(t) = h'_m(t)$, $m = 1 : M - 1$ to calculate the left hand side of the equation 7.

4. POLYNOMIAL-PHASE FOURIER KERNEL

In [18] it was demonstrated that the estimation accuracy is inversely proportional to the kernel-to-signal correlation. Therefore maximising the correlation should improve the accuracy and since the signal is modeled as a non-stationary sinusoid, a natural choice for kernel function would be the same as the model. The proposed kernel function follows:

$$\Psi_G(t) = e^{G(t)}, \quad (9)$$

where $G(t)$ is a purely imaginary polynomial of order M : $G(t) = j \sum_{m=1}^M g_m t^m$. Note that $g_0 = 0$, as any non-zero value would introduce bias in the phase estimation. From scheme 8 it is clear that an $(M - 1)$ -th degree time derivative of the kernel function is required. In the specific case of the polynomial-phase Fourier kernel the following scheme similar to 8 can be used in order to calculate the kernel function time derivatives:

$$\begin{aligned} &\Psi'_G = G' \Psi_G \\ &\swarrow \quad \searrow \\ &\Psi''_G = G'' \Psi_G + G' \Psi'_G \\ &\swarrow \quad \searrow \quad \swarrow \quad \searrow \\ &\Psi'''_G = G''' \Psi_G + 2G'' \Psi'_G + G' \Psi''_G, \\ &\vdots \end{aligned} \quad (10)$$

The main advantage of such algorithm is less restricted kernel, thus the selection of $h_m(t)$ functions can therefore be matched with an appropriate kernel functions to maximize correlation and avoid accuracy deterioration in the case of extreme parameter values.

The algorithm should initially be invoked with $G(t) = j\hat{\omega}t$, where $\hat{\omega}$ is a frequency of the magnitude spectrum peak. This yields an initial estimate of the polynomial $R(t)$: $\hat{R}(t) = \sum_{m=1}^M \hat{r}_m t^m$. This initial run of the algorithm is identical to generalized reassignment as described in [19]. In the second iteration the kernel function can be adapted to the signal by setting $G(t) = j\Im(\hat{R}(t)) = j \sum_{m=1}^M \hat{q}_m t^m$.

From 8 and 10 the following linear system of equations can be directly deduced:

$$\begin{aligned} \langle s, \Psi_G w \rangle &= \langle s, \Psi'_G w \rangle + \langle s, \Psi_G w' \rangle & \dots \\ \langle st, \Psi_G w \rangle &= \langle st, \Psi'_G w \rangle + \langle st, \Psi_G w' \rangle & \dots \\ \langle st^2, \Psi_G w \rangle &= \langle st^2, \Psi'_G w \rangle + \langle st^2, \Psi_G w' \rangle & \dots \\ \vdots & & \ddots \end{aligned} \quad (11)$$

Of a particular interest is the term written in bold, $\langle st, \Psi_G w \rangle$. When the kernel $\Psi_G(t)$ closely matches the target signal $s(t)$ then the product $\Psi_G(t)s(t) \approx 1$ and the following can be deduced:

$$\langle st, \Psi_G w \rangle = \int ts(t) \bar{\Psi}_G(t) w(t) dt \approx \int tw(t) dt. \quad (12)$$

For any symmetric window function $w(t)$ and $t \in [-\frac{T}{2}, \frac{T}{2}]$ (T being its essential time support) the above expression is very close to 0. Such cases occur when the signal exhibits low or no amplitude modulation causing the linear system of equations close to singular, rendering the algorithm essentially useless. Such a drawback can simply be avoided by artificially inducing some amplitude modulation into the signal and then subtracting it from the estimate obtained. A very small amount of the amplitude modulation of magnitude around 10^{-10} is sufficient to stabilize the system and significantly improve the estimates.

5. RESULTS

The tests conducted were identical to those in [19]. The metric used was the signal to residual ratio (SRR):

$$SRR = \frac{\sum_i h_i s_i^2}{\sum_i h_i (s_i - \hat{s}_i)^2}, \quad (13)$$

where $s_i, i = 1..N$ are samples of the original signal $s(t)$ (without noise), $\hat{s}_i, i = 1..N$ are the samples of the estimated signal and $h_i, i = 1..N$ are samples of the weighting function - Hanning window. A model degree of 3 was chosen and the Hanning² function of length 1024 was used as the window function. The test signals analyzed were real sinusoids sampled at 44100Hz. The parameters of the test sinusoids were varied in the following way: 10 phase values in the $[0, 0.45]\pi$ interval, 10 linear log-amplitude modulation values in the $[0, 0.0045]$ /frame interval (roughly corresponds to the $[0, 200]$ /s interval), 10 frequency values in the $[10.982, 11.021]$ Hz) and 10 linear frequency modulation values in the $[0, 27]$ bins/frame interval (roughly corresponds to the $[0, 16.000]$ Hz/s). The tests were conducted in 3 separate groups for the original reassignment (labeled **GEN RM**) and the one using the polynomial-phase kernel (labeled **GEN RM PPT**). In group 1 (figure 1), the linear frequency modulation was set to 0 while the log-amplitude modulation was varied (x-axis) in the mentioned range. In group 2 (figure 2) the log-amplitude modulation was set to 0 while the linear frequency modulation was varied (x-axis) in the mentioned range. In group 3 (figure 3), both the FM and log-AM were jointly varied (x-axis) in **double the range** compared to the groups 1 and 2. In the first part (labeled **SNR: Inf dB** in the plots) no noise was added to the signal and in the second part (labeled **SNR: 0dB** in the plots) a Gaussian white noise of the energy equal to that of the clean signal was added. The range of the log-AM/FM for group 3 was doubled intentionally to examine properties of both algorithms in highly modulated cases. The frequency range was selected around half of Nyquist frequency in order to avoid self-interference.

As predicted, in the noiseless case the proposed kernel greatly diminishes the effect of the frequency modulation on the parameter estimation accuracy. For FM only case (figure 2), the kernel adaptation procedure leaves the accuracy completely unaffected even for very high FM values. On the other hand, the presence of AM does affect the accuracy slightly, as can be seen in the figures 1 and 3, yet the improvement over the original method is significant. In the **SNR: 0dB** case, the performance is almost indistinguishable to the one of the original generalized reassignment.

6. CONCLUSION AND FUTURE WORK

In this paper, an improvement of the generalized reassignment method was described. The main idea of the improvement is the use of an adaptive polynomial-phase Fourier kernel in conjunction with the general reassignment algorithm. The algorithm exhibits a significant improvement in accuracy compared to the original method in the case of clean signal, as the effect of frequency modulation is minimized by the adaptive kernel. For a stationary sinusoid, the accuracy is comparable to the original method, however an increase in accuracy is observed in the case of non-stationary ones, reaching almost 50dB in the most modulated case (group 3). The method does not improve the analysis of the original algorithm if 0 dB Gaussian white noise added. The reason for this is the kernel adaptation works in the opposite way to which is desired. This

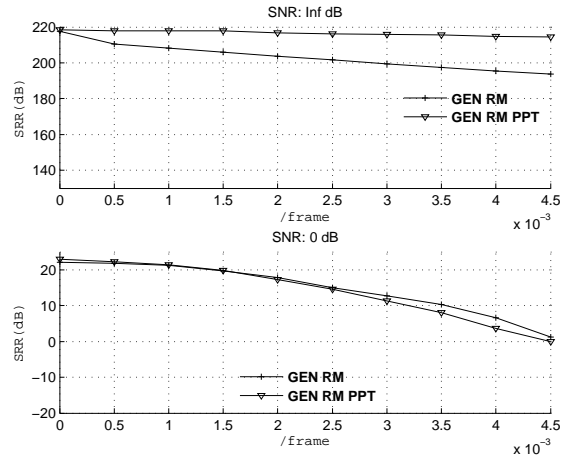


Figure 1: Group 1 (AM only)

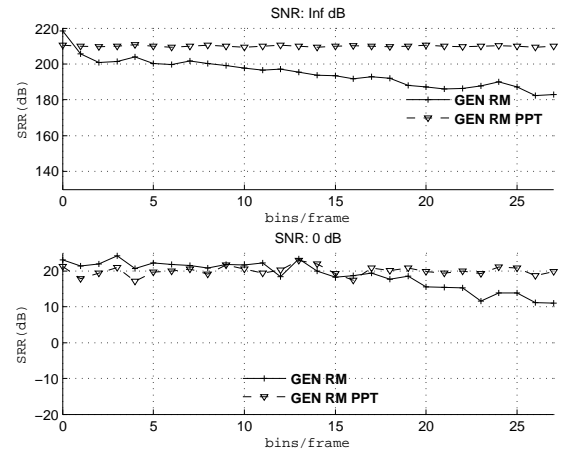


Figure 2: Group 2 (FM only)

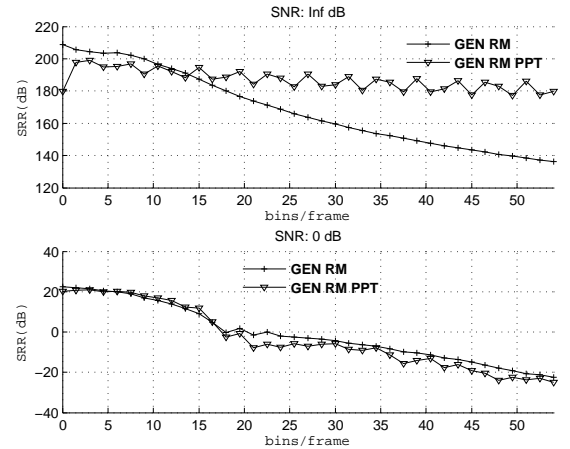


Figure 3: Group 3 (AM and FM)

is because it uses the estimate of the original method, which is not precise enough at such a high noise level, therefore the error in the

input parameters corrupts the final estimate.

In group 3, the most modulated case corresponds to 32.000Hz/s change. This may seem excessive for analyzing real world music related signals. However, a higher order modulation polynomials could exhibit even larger linear FM values, as its contribution can be canceled or balanced out by the second or higher order terms. So as the kernel is adapted to the sinusoid in question, the energy concentration of its representation in the transform domain is increased: the bandwidth of the non-stationary sinusoid is reduced. This is a desirable property in the case of multicomponent signals, where side-lobes of a sinusoid cause significant interference to the neighboring partials.

All the measured tests were conducted with Hanning² window, which would substantially increase interference in a multicomponent scenario, as its main lobe is wider than that of the Hanning window. An attempt to construct an L^2 window function with a lower bandwidth should receive some attention, allowing an improvement of the method using a model degree of up to 4.

As already mentioned in the previous section, the frequencies under study were varied around half of the Nyquist, therefore the signal self-interference was minimized. Since the nature of the interpartial interference does not resemble that of a Gaussian white noise, the results presented here cannot be generalized to a multicomponent cases, thus an assessment of the method's accuracy in such cases should be conducted.

The algorithm was designed in such a way that it can be iteratively ran as many times as desired, which raises a question of the convergence in a noisy case. Preliminary tests suggest, that such iteration converges and improves the result as long as the initial estimates don't deviate too much from the true values. Further experiments are required to further define the region of convergence.

7. REFERENCES

- [1] X. Serra, *A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition*, Ph.D. thesis, Stanford University, 1989.
- [2] J. Bonada, "Wide-band harmonic sinusoidal modeling," in *Proc. of 11th Int. Digital Audio Effects (DAFx-08)*, Espoo Finland, Sept. 2008.
- [3] R. McAulay and T. Quatieri, "Speech analysis/Synthesis based on a sinusoidal representation," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 4, pp. 744 – 754, Aug. 1986.
- [4] J. Bonada, O. Celma, A. Loscos, J. Ortola, and X Serra, "Singing voice synthesis combining excitation plus resonance and sinusoidal plus residual models," in *Proc. of International Computer Music Conference 2001 (ICMC01)*, La Habana, Cuba, Sept. 2001.
- [5] S.W. Hainsworth, M.D. Macleod, and P.J. Wolfe, "Analysis of reassigned spectrograms for musical transcription," in *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, New Paltz, New York, Oct. 2001, pp. 23–26.
- [6] K. Brandenburg, J. Herre, J. D. Johnston, Y. Mahieux, and E. F. Schroeder, "Aspec-adaptive spectral entropy coding of high quality music signals," in *Audio Engineering Society Convention 90*, Feb. 1991, vol. 90, p. 3011.
- [7] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *Signal Processing, IEEE Transactions on*, vol. 43, no. 5, pp. 1068–1089, 1995.
- [8] M. Desainte-Catherine and S. Marchand, "High-precision fourier analysis of sounds using signal derivatives," *J. Audio Eng. Soc.*, vol. 48, no. 7/8, pp. 654–667, Sept. 2000.
- [9] S. Marchand, "Improving spectral analysis precision with an enhanced phase vocoder using signal derivatives," in *Proc. of the Digital Audio Effects Workshop*, Barcelona, Spain, Nov. 1998, pp. 114–118.
- [10] L. Cohen and C. Lee, "Standard deviation of instantaneous frequency," in *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, Glasgow, Scotland, 1989, vol. 4, pp. 2238–2241.
- [11] L. Cohen, *Time Frequency Analysis: Theory and Applications*, Prentice Hall PTR, facsimile edition, Dec. 1994.
- [12] K. L. Davidson and P. J. Loughlin, "Instantaneous spectral moments," *Journal of the Franklin Institute*, vol. 337, no. 4, pp. 421–436, July 2000.
- [13] K. Kodera, R. Gendrin, and C. Villedary, "Analysis of time-varying signals with small BT values," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 64–76, 1978.
- [14] Axel Röbel, "Estimating partial frequency and frequency slope using reassignment operators," in *Proc. of The International Computer Music Conference 2002 (ICMC02)*, Gothenburg, Sweden, Sept. 2002, pp. 122–125.
- [15] M. Abe and J.O. Smith, "AM/FM rate estimation for time-varying sinusoidal modeling," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, Mar. 2005, vol. 3, pp. iii/201–iii/204.
- [16] S. Marchand and P. Depalle, "Generalization of the derivative analysis method to Non-Stationary sinusoidal modeling," in *Proc. of 11th Int. Digital Audio Effects (DAFx-08)*, Espoo, Finland, Mar. 2008.
- [17] X. Wen and M. Sandler, "Evaluating parameters of time-varying sinusoids by demodulation," in *Proc. of the 11th Int. Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sept. 2008.
- [18] M. Betser, "Sinusoidal polynomial parameter estimation using the distribution derivative," *IEEE Transactions on Signal Processing*, vol. 57, no. 12, pp. 4633 – 4645, Dec. 2009.
- [19] X.Wen and M. Sandler, "Notes on model-based non-stationary sinusoid estimation methods using derivative," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 2009.
- [20] S. Mušević and J. Bonada, "Comparison of non-stationary sinusoid estimation methods using reassignment and derivatives," in *Proc. of 7th Sound and Music Computing Conference*, Barcelona, Spain, July 2010.

SIGNAL RECONSTRUCTION FROM STFT MAGNITUDE: A STATE OF THE ART

Nicolas Sturmel*, Laurent Daudet

Institut Langevin
ESPCI CNRS UMR 7587
10 rue Vauquelin 75005 Paris, France
firstname.lastname@espci.fr

ABSTRACT

This paper presents a review on techniques for signal reconstruction without phase, i.e. when only the spectrogram (the squared magnitude of the Short Time Fourier Transform) of the signal is known. The now standard Griffin and Lim algorithm will be presented, and compared to more recent blind techniques. Two important issues are raised and discussed: first, the definition of relevant criteria to evaluate the performances of different algorithms, and second the question of the unicity of the solution. Some ways of reducing the complexity of the problem are presented with the injection of additional information in the reconstruction. Finally, issues that prevents optimal reconstruction are examined, leading to a discussion on what seem the most promising approaches for future research.

1. INTRODUCTION

The ubiquitous Short Time Fourier Transform (STFT) is a very efficient and simple tool for audio signal processing, with a representation of the signal that simultaneously displays both its time and frequency content. The STFT computation is perfectly invertible, fast (based on the Fast Fourier Transform (FFT)), and provides a linear framework well suited for signal transformation. However, a majority of these modifications act on the magnitude of the STFT ; in this case phase information is lost, or at least corrupted. Source separation, for instance, is often based on the estimation of the time-frequency local energy of the sources, and the isolated sources are usually recovered through Wiener filtering [1], i.e. with the phase of the original mixture. Other cases of adaptive filtering, like denoising [2], usually perform subtraction in the amplitude domain, once again not taking account of the phase of the signal. Signal modifications, such as time-stretching or pitch shifting [3], may also involve changes on the magnitude of the STFT (adding/removing frames, moving bins) without perfect knowledge of the expected structure of the phase. Although phase vocoder [4] brings some answers to the problem, the overall quality of the modification is still perfectible.

Furthermore, accurate reconstruction of a signal from its magnitude STFT is also of paramount importance in the domain of signal representation. Many works are addressing the relation between magnitude and phase of a Discrete Fourier Transform (DFT) [5, 6, 7]. Therefore, solving convergence issues of existing algorithms could also give ways of solving the problem of phase and magnitude dependency in the time-frequency domain. In short,

being able to reconstruct a signal while only knowing its magnitude could bring significant improvements in many situations from source separation to signal modification.

Here, the key point is that the STFT has an important property: redundancy of the information. For a real signal, each length- N analysis window provides $N/2 + 1$ independent complex coefficients (keeping only components corresponding to positive frequencies), and with the additional constraint that the coefficients at frequencies 0 and $N/2$ are real by construction, this amounts to N real coefficients (in other words, the Discrete Fourier Transform is an orthogonal transform). However, with the STFT the analysis is always carried out with an overlap between adjacent analysis windows. In the case of minimal overlap of 50%, a real input signal of length N provides $2N$ real coefficients (neglecting here boundary effects). In the common case where the overlap is higher than 50%, this redundancy of information gets even higher. Similarly, the FFT can be oversampled in frequency (with zero-padding in time), providing more coefficients per frame.

This brings an important point: the STFT has to verify a so-called “consistency criterion” [8]. In other words, the set of complex STFT coefficients lives within a subset of the space $\mathbb{C}^{N \times M}$, but is not isomorphic to it: in general, an array of complex coefficients does not correspond to the STFT of a signal. Now, when keeping only the magnitude of the STFT, a real input signal of length N provides $N + 1$ real coefficients (with 50% overlap): phase reconstruction from magnitude-only spectrograms *may* still be possible [3]. The main issue is whether some crucial information has been lost by taking the magnitude, bringing ambiguities and/or ill-posedness issues. In the case of source separation, for instance, Gunawan showed [9] that phase reconstruction improved the quality of the separation. In the case of adaptive filtering, Le Roux showed [10] that the inconsistency criterion led to an improved estimation of the Wiener filter.

The goal of this article is to provide a state of the art in the problem of signal reconstruction from spectrograms (the squared magnitude of the STFT). Its goal is not only to review the benefits and drawbacks of each of the published methods, but also to discuss fundamental and sometimes open issues that make this problem still very active after decades of intense research. The article is organized as follows: the framework of the STFT will be presented in section 2, and the unicity of the representation will be discussed in section 3. The baseline technique for phase reconstruction, the so-called Griffin and Lim algorithm, will be presented in section 4 and quantification of the convergence will be discussed in section 5. Then, more recent reconstruction techniques will be presented: blind reconstructions in section 6 and informed ones in section 7. Finally, issues that arise when trying to achieve perfect reconstruction

* This work was supported by the DReaM project (ANR-09-CORD-006) of the French National Research Agency CONTINT program.

tion of the signal will be discussed in section 8 and applications of such phase estimation to digital audio processing in section 9 prior to the conclusion of the document in section 10.

2. SHORT TIME FOURIER TRANSFORM

Let $x \in l_2(\mathbb{R})$ be a real, discrete signal, of finite support. On this support, we define the *STFT* operator such that $S(n, m) = STFT[x]$ computed with an analysis window w of length N and an overlap $N - R$ (i.e., a hop size of R samples between consecutive analysis windows):

$$S(n, m) = \sum_{k=0}^{N-1} e^{-i2\pi \frac{kn}{N}} w(k)x(k + Rm) \quad (1)$$

Here, n is the frequency index, and m the time index. Inversion of this *STFT* is achieved by the synthesis operator $STFT^{-1}$ described in equation (2) using the synthesis window s which gives the signal \tilde{x} :

$$\tilde{x}(l) = \sum_m s(l - mR) \sum_n S(n, m) e^{i2\pi n \frac{l-mR}{N}} \quad (2)$$

If the synthesis and analysis windows verify the energy-complementary constraint:

$$\sum_m w(l + mR)s(l + mR) = 1$$

then perfect reconstruction is achieved: $\tilde{x} = x$.

However, one might want to have more freedom in the choice of analysis / synthesis windows, and therefore the $STFT^{-1}$ operator must include a window ponderation such that $\tilde{x}(l) = \frac{1}{\tilde{s}(l)} STFT^{-1}[S]$, where $\tilde{s}(l) = \sum_m w(l+mR)s(l+mR)$ which is equivalent, up to boundary effects, to constraining the synthesis window to $\frac{s(l)}{\tilde{s}(l)}$. In [11], the inverse STFT is also described with the use of a vector formulation.

The different domains involved and the functions used to pass from one to another are presented on figure 1. The spectrogram W is the squared magnitude¹ of S and is given by $W = SS^*$ where S^* is the complex conjugate of S . Note that the spectrogram of a signal is also its autocorrelation and can be used as such for the interpolation of signals [12]. W is a set of real non-negative numbers $\in \mathbb{R}_+^{N \times M}$. The goal of the reconstruction is then to estimate $\tilde{S}(n, m)$ such that $\tilde{S} \in \mathbb{S}_{N,M}$, where $\mathbb{S}_{N,M}$ is the subset of $N \times M$ complex arrays representing co-called “consistent” STFTs, while keeping $\tilde{S}S^* = W$. Consistency of S is provided by the constraint $\mathcal{I}(S) = 0$, where \mathcal{I} is defined by:

$$\mathcal{I}(S) = S - STFT[STFT^{-1}[S]] \quad (3)$$

In many applications such as the ones mentioned in the introduction, the array W used for reconstruction might not itself belong to the set of “consistent spectrograms” (the image of $\mathbb{S}_{N,M}$ by the operator $M \rightarrow |M|^2$). This might be due to the fact that the estimation of W is corrupted by noise (for denoising), or the cross-talk of other sources (for source separation), or because W is obtained through an imperfect interpolation algorithm (for time-stretching). In this case, there is no signal x that exactly verifies

¹It should be noticed that some authors alternatively refer to spectrogram as the set S , i.e. the complex STFT coefficients

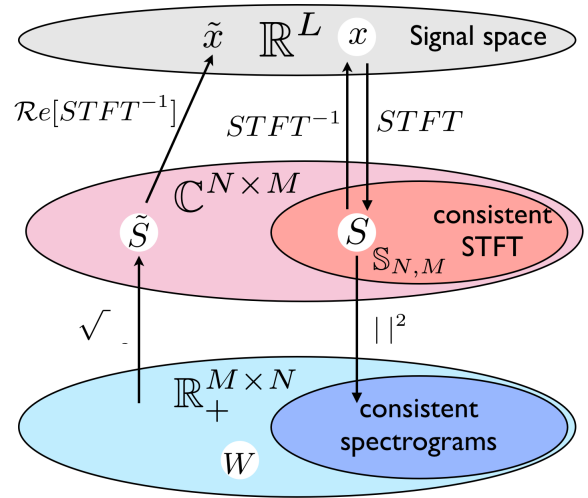


Figure 1: Domains involved when processing STFTs and spectrograms (expanded from [8]).

$S_x S_x^* = W$. There, the goal is to find the closest approximation, that minimizes the norm of $\mathcal{I}(S)$ (for some matrix norm, usually the Froebenius norm). In other words, one looks for the set $\mathbb{S}_{N,M}$ of consistent STFTs that verify $SS^* = W$.

Because we are specifically addressing a problem that uses compact STFTs, we discard techniques involving oversampling of each DFT [13, 14]: oversampling the DFT, while retaining the overlapping of the frames, introduces a redundancy of information that is too large to be handled in most practical cases. Signal reconstruction in those conditions can be considered solved by the previous studies even in the case of an isolated frame [15]. In this review, we will focus on techniques that, on the contrary, do not require specific constraints on the window design, the DFT oversampling, or hop size (we just assume that the STFT and inverse STFT are fixed and well-defined).

When trying to estimate the phase of an STFT from its magnitude only, some problems arise: the unicity of the representation [16, 12] discussed in section 3, how to quantify the convergence of the reconstruction (section 5), but also the tendency of reconstruction algorithms to catch local, non optimal, minima. A notable issue preventing optimal convergence is the so-called stagnation of the optimization [17] and will be discussed in section 8.

3. UNICITY OF THE REPRESENTATION

When addressing the problem of perfect reconstruction of a signal from its spectrogram, the first question that comes in mind is the unicity of the representation: can two different signals provide the same spectrogram? The work of Nawab [12] produced some practical answers to the problem while only providing sufficient but not mandatory conditions to guarantee the unicity of x represented by $W(n, m)$. Some other works, such as [16] addressed signal uniqueness with the use of asymmetric windows ($w(n) \neq w(N - n)$), but such window is not suited for analysis of the spectrogram for the sake of phase linearity amongst other causes.

3.1. Sign indetermination

Some simple examples can be given to prove that unicity is not always verified. This is caused by the sign indetermination $|STFT[x]| = |STFT[-x]|$. Take for instance two signals x_1 and x_2 such as they do not overlap: $x_1 = 0$ outside $[N_{1A}; N_{1B}]$ and $x_2 = 0$ outside $[N_{2A}; N_{2B}]$ with $N_{1B} + N < N_{2A}$, then $x_1 - x_2$ and $x_1 + x_2$ have the same STFT $S(n, m)$.

Therefore, there are at least two signals x and $-x$ verifying the spectrogram W and the solution can only be unique under some constraints such as positivity of the signal (for instance in the case of image processing). But when this sign indetermination happens between big chunks of an audio signal, this case is either perceptually insignificant or can be countered by some simple knowledge on the signal.

However, it will be shown that this sign problem can happen locally in the reconstructed signal and regardless of its structure, this phenomenon is called *stagnation* by Fienup et al. [17] and will be discussed in section 8.

3.2. Conditions for the unicity of the reconstruction

The important conditions providing unicity in the case of a partial overlap, that is when hop size is $R > 1$, are given by Nawab [12]:

1. Known window function $w(n)$
2. Overlap of at least 50% ($R \leq \frac{N}{2}$).
3. Non zero window coefficients within the interval $[0; N]$
4. One sided signal, to define at least one boundary
5. Knowing R consecutive samples of the signal to be reconstructed starting from the first non-zero sample.
6. Less than R consecutive zeros samples in the signal.

Condition 1 of knowing $w(n)$ can be simply explained. This was illustrated by Le Roux in [18], with the example of designing an inconsistent STFT $H \in \mathbb{C}^{N \times M}$ so that $\sum |H| > 0$ but $STFT(STFT^{-1}(H)) = 0$ only for a given analysis/synthesis window pair. Since each analysis window has a different time-frequency smearing (see figure 6, in section 6), the information contained in the spectrogram is directly linked to w . This is especially true for inconsistent STFTs, of which the spectrogram is a particular case.

Condition 2 suggests that the amount of data contained by $|S(n, m)|$ is superior or equal to the one originally present in x , while condition 3 prevents missing informations due to zeros in w . Without any a priori on the signal, necessity of those two conditions seems rather natural. Enforcing regularity on the signal (like the techniques discussed in section 7) can lower those specific conditions.

Condition 4 imposes boundaries to the signal, allowing injection of some informations for the reconstruction, similar to the approach of Hayes and Quatieri in [19]. These boundaries were also used by Fienup et al. [17] but the support of an audio signal is too big in regard to the analysis window in order for such condition to be efficient. In fact, much more happens between the boundaries.

Since Nawab's work was based on successive interpolation of the signal, conditions 5 and 6 were established in order to know precisely the first R samples of the signal and continuously interpolating the signal without gaps. We feel that condition 5 is not always necessary, but condition 6 prevents sign indetermination problems like illustrated in section 3.1.

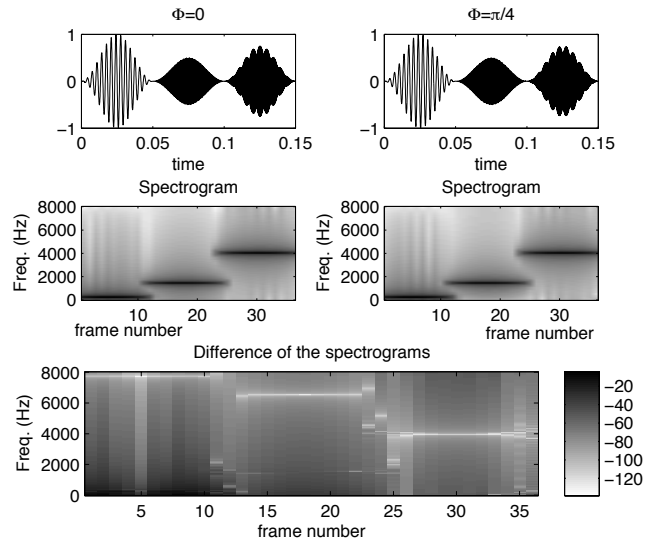


Figure 2: Spectrogram differences between two simple signals x_0 and $x_{\pi/4}$.

Some examples will be given throughout the paper in order to show that if the signal is not unique, it often comes down to the duality of the sign indetermination. We will also show in section 8 that greater issues are preventing the reconstruction and that unicity of the solution can be overlooked until those issues are solved. However those issues will often be linked to the unicity problem.

3.3. Phase rotation and spectrogram invariance

One common misconception about spectrogram is that it is phase invariant. Of course, if one were to work with complex signals, this phase invariance would be verified, but whether this still holds for real signals (whatever this means) is not so obvious.

For real signals, the only way to appropriately define the phase of the signals is within the framework of analytic signals. Let us assume that the signal x under study is the real part of a mono-component analytic signal H with slowly-varying amplitude $A(t)$: $x(t) = \text{Re}(H) = A(t) \cos(\omega t)$, and let us construct the families of functions x_Φ for the same amplitude A and frequency ω , but with varying absolute phase Φ : $x_\Phi = \text{Re}[H e^{i\Phi}]$. If phase invariance were to hold, the spectrogram $|S_\Phi|^2$ of x_Φ would be the same as $|S_0|^2$ for any value of Φ .

Figure 2 shows the signal, spectrogram and absolute spectrogram difference of x_Φ for $\Phi = 0$ (left) and $\Phi = \frac{\pi}{4}$ (right) for three frequencies (300, 1500 and 4050Hz) at 16kHz sampling frequency and for an envelop A in the shape of a Hanning window with three different amplitudes ($1, \frac{1}{2}$ and $\frac{3}{4}$). The difference is computed as $||S_0| - |S_{\pi/4}||^2$. As one can see, this difference has an energy far from negligible.

Two interesting remarks can be made: first, the error is spread throughout the spectrum and not only in the vicinity of the signal's frequency. Second, this error is not either concentrated in time around the onset or offset of the tones: it can be shown as well that there is a similar error even when the amplitude of the signal stays constant.

Figure 3, shows the average spectrogram difference $\mathcal{C}(S_0, S_\Phi)$

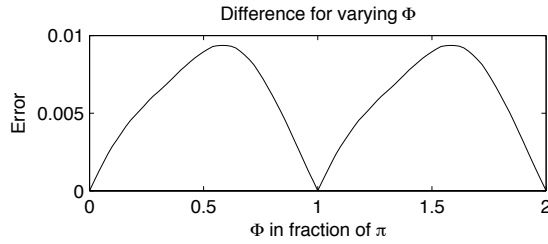


Figure 3: Spectrogram differences (equation (4)) for varying Φ in x_Φ .

as defined by

$$\mathcal{C}(S_0, S_\Phi) = \sqrt{\frac{\sum_{n,m} | |S_0(n,m)| - |S_\Phi(n,m)| |^2}{\sum_{n,m} |S_0(n,m)|^2}} \quad (4)$$

for varying Φ from 0 to 2π . One can see that the difference is π periodic due to the sign indetermination of $|S_\Phi(n,m)|$ and that most of the time it is inferior to 0.01 (i.e. -20dB).

This small experiment leads to the following rule of thumb: strictly speaking, the STFT is *not* phase invariant. However, when the computation is only made with low precision (less than 20 dB), the standard error criteria on the spectrogram don't "see" the phase. When minimizing this error, it appears that the original signal is indeed the true minimum but within a very flat surface. However, this fact that STFT is not strictly invariant to phase is good news: phase information seems to be present to some extent in the amplitude, but as a second-order effect. We shall see that this observation is the basis for discussion on the main issues making phase reconstruction such an intricate problem.

3.4. Perfect reconstruction

While the signal to be reconstructed from W is not necessarily unique, our goal is to find the most accurate reconstruction in regard to the original signal x . We call perfect reconstruction the estimation of the signal \tilde{x} with an error of at most the measure error on x . If x is 16bits sampled, then the error power to achieve is approximatively equal to the quantification error power, that is to say approx. -90dB.

Moreover, we will consider perfect reconstruction as the estimation of x or $-x$. That is, we are implicitly discarding the global sign problem in the determination of x . We will show in section 8 that local indetermination of this sign can cause convergence issues.

4. ITERATIVE RECONSTRUCTION OF THE SIGNAL: THE GRIFFIN AND LIM BASELINE ALGORITHM

Based on the Gerchberg and Saxon algorithm [20], Griffin and Lim proposed the first global approach to solve the problem of signal reconstruction from spectrograms [3]. Due to the good perceptual results despite its simplicity for a basic implementation, this reconstruction algorithm remains the baseline for all subsequent work. Note that, as in the case of Gerchberg and Saxon reconstruction of the phase, uniqueness of the reconstruction is not guaranteed.

The approach from Griffin and Lim relies on a two-domain constraint, similar to the work of Hayes [15]. Before reconstruction, the spectrogram W of the STFT S is known but the phase

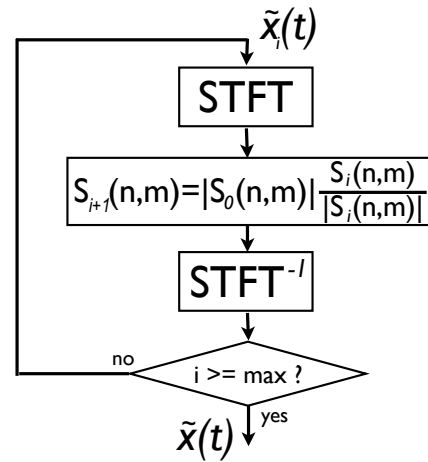


Figure 4: The iterative framework of Griffin and Lim [3]

$\angle S$ is unknown and can be initialized to 0 or at random values. In the spectral domain, absolute values of the estimated STFT \tilde{S}_i are constrained to $|S_0| = \sqrt{W}$ at each iteration i , while the temporal coherence (as defined by equation (3)) of the signal is enforced by the operator $STFT[STFT^{-1}]$.

The algorithm is presented on figure 4. First, it is initialized with $S_0 = \sqrt{W}$. At iteration i , the estimated STFT \tilde{S}_i is computed and $\angle \tilde{S}_i$ is given to the original spectrogram so that the resulting time domain signal x_i is computed by inverse STFT of $|S_0| \frac{\tilde{S}_i}{|\tilde{S}_i|}$. In [3] it is shown that the mean square error between the STFT of the signal x_i and the estimated STFT of amplitude $|S_0|$ can be expressed as a distance:

$$d(S_0, \tilde{S}_i) = \sum_{n,m} | |S_0| \frac{\tilde{S}_i}{|\tilde{S}_i|} - \tilde{S}_i |^2 \quad (5)$$

and can be reduced to:

$$d(S_0, \tilde{S}_i) = \sum_{n,m} | |S_0| - |\tilde{S}_i| |^2 \quad (6)$$

It is also demonstrated that the gradient of d verifies $\Delta d(S_0, \tilde{S}_i) \leq 0$ and that this technique therefore reduces the distance d at each iteration.

This algorithm presents three main drawbacks:

1. First, its computation requires offline processing, as it involves computation of the whole signal at each iteration, and computation of both an STFT and an inverse STFT.
2. Second, convergence can be very slow, both in terms of computation time per iteration and by the number of iterations before convergence.
3. Finally, the algorithm does not perform local optimization to improve signal consistency, neither does it provide a consistent initialization of the phase from frame to frame.

Griffin and Lim's algorithm often provides time-domain signals that sound perceptively close to the original. However, depending on the sound material and the STFT parameters, some artifacts can be perceived: extra reverberation, phasiness, pre-echo... Indeed, while looking at the temporal structure of the reconstructed signals, we can see that they are often far enough from

the original to produce RMS error above 0dB. Although the corresponding sound quality may be sufficient in many cases, there are some application scenario where this may be a severe limitation. For instance, in the context of audio source separation, one may want to listen to the residual signal without the estimated source (karaoke effect): obviously a badly estimated time-domain signal prevents a correct source subtraction from the mix.

5. CONVERGENCE CRITERION

In order to assess the performance of the reconstruction, different criteria have been proposed. The most common ones are:

1. The spectral convergence \mathcal{C} , expressed as the mean difference of the spectrogram W with the absolute value of the reconstructed STFT \tilde{S} as expressed by:

$$\mathcal{C} = \sqrt{\frac{\sum_{n,m} |\sqrt{W(n,m)} - \sqrt{\tilde{S}(n,m)\tilde{S}^*(n,m)}|^2}{\sum_{n,m} W(n,m)}} \quad (7)$$

The convergence criterion \mathcal{C} relates directly to the minimization process of Griffin and Lim's technique (equation (6)). This is the distance between the current coherent spectrogram and the target spectrogram. Then, when $\mathcal{C} = 0$, perfect reconstruction is achieved modulo unicity of the solutions.

2. The consistency \mathcal{I} of the estimated STFT \tilde{S} as given in equation (3). Again, $\mathcal{I} = 0$ means an accurate reconstruction, up to invariants.
3. The signal x to reconstruction \tilde{x} root mean square error power:

$$\mathcal{R} = \sqrt{\frac{\sum (x(n) - \tilde{x}(n))^2}{\sum x(n)^2}} \quad (8)$$

This criterion, analogous to the inverse of the signal-to-noise ratio, gives a better view of the reconstruction quality (we chose error over signal-to-noise ratio in order to observe the variations of \mathcal{C} and \mathcal{R} in the same direction). Note that the computation of \mathcal{R} requires the knowledge of the original signal x . Therefore, it can only be used in (oracle) benchmarking experiments, and not in (blind) practical estimation. In this case, when $\mathcal{R} = 0$ the reconstruction is strictly equal to the original.

Obviously, the choice of the convergence criterion will have an effect on the discussion of the results obtained by each method. Even if $\mathcal{R} = 0$ is equivalent to $\mathcal{C} = 0$, one can easily find very small values of \mathcal{C} associated with high values of \mathcal{R} .

Such issue is illustrated on figure 5 in the simple case of the DFT. The signal x used to compute figure 5 is a speech signal sampled at 16kHz and quantized on 16 bits. A random phase delay $\Phi(n)$ is computed, respecting the Hermitian symmetry ($\Phi(-n) = -\Phi(n)$), and making sure that this delay is always an integer in samples $\forall n$. Then, the phase of the DFT of x is shifted by $\Phi(n)$, multiplied with an integer factor k , with k ranging from 1 to 20. This is done through

$$\tilde{X}_k(n) = X_k(n)e^{ik\Phi(n)}$$

The resulting time-domain signal is called \tilde{x}_k . The two signals x and \tilde{x}_k have the same energy ($XX^* = \tilde{X}_k\tilde{X}_k^*$), but are randomly delayed across frequencies. The figure displays the convergence criterion ($20 \log \mathcal{C}$) and the reconstruction error ($20 \log \mathcal{R}$)

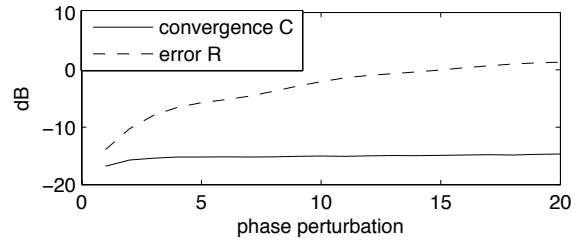


Figure 5: Difference between the \mathcal{C} and \mathcal{R} criteria used to evaluate the signal reconstruction, as a function of the amplitude of a random delay (integer in samples) on the DFT spectrum.

both in dB between signals x and \tilde{x}_k . Since there are two possible solutions (x and $-x$), \mathcal{R} displayed on figure 7 is computed as $\min(\mathcal{R}|_x, \mathcal{R}|_{-x})$. In this figure, one can see that the two criterions evolve separately. While \mathcal{C} is staying at approx. -14dB , \mathcal{R} is slowly rising to values above 0dB. This illustrates the fact that \mathcal{C} may not be a good indicator of the reconstruction quality, with respect to the original signal.

6. BLIND TECHNIQUES FOR SIGNAL RECONSTRUCTION

In this section, we review recent techniques that have been designed to improve Griffin and Lim's algorithm.

6.1. STFT consistency

STFT consistency of equation (3) can lead to the spectral domain only formulation of Griffin and Lim's least square estimation of the signal. In [8], an extensive work is presented to show how equation (3) can be used for the estimation of the phase of the corresponding coherent STFT. For instance, equation (10) gives a phase estimate Φ at each coordinate (n, m) of the STFT:

$$\begin{aligned} \mathcal{I}(n, m) &= S(n, m) - STFT[STFT^{-1}[S(n, m)]] \\ \mathcal{I}(n, m) &= \sum_{p=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{q=1-Q}^{Q-1} e^{i2\pi \frac{qn}{Q}} \alpha(p, q) S(n-p, m-q) \quad (9) \\ \Phi(n, m) &= \angle(S(n, m) + \mathcal{I}(n, m)) \quad (10) \end{aligned}$$

with $\alpha(p, q) = -\frac{1}{N} \sum_k \frac{w(k)s(k)}{\tilde{s}(k)} e^{-i2\pi p \frac{k+qR}{N}} + \delta_p \delta_q$

The term $\alpha(p, q)$ is the convolutive kernel applied to the STFT, that ensures both time domain (coordinate q) and frequency domain (coordinate p) coherence of the representation (this is the equivalent of the so-called "reproducing kernel" in wavelet analysis). This kernel is directly computed with the analysis and synthesis windows, and is invariant for the whole STFT. The shape of different kernels $\alpha(p, q)$ is given on figure 6 for four different window functions. The temporal dispersion of the kernel has a weak dependency on the window shape, but the frequency distribution is in direct relation to the spectral leakage of the window function [21].

The expression of $\mathcal{I}(n, m)$ given by equation (9) makes explicit the consistency criterion given in equation (3). This criterion is particularly efficient to provide information on the local coherence of the STFT as the phase correction depends directly on the value of $\frac{|\mathcal{I}(n, m)|}{|S(n, m)|}$. Equation 10 is also the direct application of

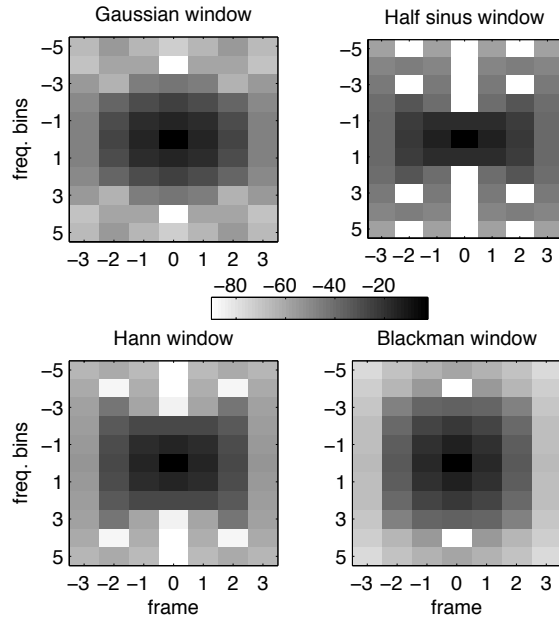


Figure 6: The influence of different windows on the STFT representation for an overlap of 75%. (amplitudes in dB)

Griffin and Lim optimization and follows the convergence of distance d defined in equation (6).

Additional studies in the same line [8] proposed solutions to lower the computation time while keeping a similar convergence speed. First, limiting the frequency domain span of the window α drastically lowers computation time while introducing only minimum error. When using analysis windows with low spectral leakage, one can reduce the term p of equation (9) to, for instance, the range $[-2; 2]$. This simplification significantly reduces the computation time, at the cost of a small error typically below 0.1%. Figure 6 presents some shapes of $\alpha(p, q)$ for different analysis and synthesis windows. We can see that the energy is concentrated around $(0, 0)$ especially for the half sinus window (used for the experiments in [8, 22]), allowing further approximation to the frequency bins around 0.

The second simplification is the use of sparseness of the signal in the time-frequency domain, in order to only update the bins of high energy. At each iteration, bins of lower and lower energy are updated. Empirical results shows that such simplification does not significantly modify the reconstructed signal \tilde{x} at convergence, while drastically lowering computation time.

When using both simplifications, computation times given in [8] show a reduction by a factor 10 to 40 over the original Griffin and Lim iterative STFT reconstruction. This method improves convergence speed but does not significantly improve the final quality of the reconstruction. Note that both the computation time and the framework of this technique allow for real-time implementation, with minimal delay.

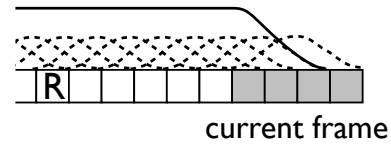


Figure 7: Real Time Iterative Spectrogram Inversion for an overlap of 75%.

6.2. Real-Time Iterative Spectrogram Inversion

The main drawback of Griffin and Lim's reconstruction algorithm is the processing of the whole signal for each iteration, preventing any use for on-line processing. Zhu and al. [23] proposed two implementations of the reconstruction, starting with a constraint of a real-time implementation.

First, the baseline Real Time Iterative Spectrogram Inversion (RTISI [24]) technique is based on the coherence of preceding reconstructions in regards to the frame begin reconstructed. This technique illustrated on figure 7, can be decomposed into two steps:

1. Consider the m -th frame S^m of the STFT $S(n, m)$ with its window function w_m and the signal \tilde{x}_m which contains the weighted sum (equation (2)) of formerly processed frames. Then, $\angle \tilde{S}_0^m$ is initialized so that:

$$\angle \tilde{S}_0^m = \angle DFT[w_m \tilde{x}_m]$$

2. Then, the iterations are done as in Griffin and Lim, but restrained to frame m . At each step:

$$\begin{aligned} \angle \tilde{S}_i^m &= \angle DFT[w_m \tilde{x}_m + w_m \tilde{x}_{i-1}^m] \\ \tilde{x}_i^m(l) &= s(l) DFT^{-1}[\angle \tilde{S}_i^m] \end{aligned}$$

This method is especially suited for multiple window length STFT, in a similar way to the window-switching method of MPEG 2/4 AAC coding [25]. However, RTISI offers results somewhat lower than Griffin and Lim's, mainly caused by the lack of look-ahead and optimization toward the *future* of the signal.

Therefore, a second method, RTISI with Look-Ahead (RTISI-LA [26]) was proposed. It is described by the scheme of figure 8. This method performs phase estimation of RTISI on k frames after the current one, ensuring that the estimated phase for the frame soon to be committed in the resulting signal \tilde{s} is both in agreement with the past and future evolutions of the signal.

Convergence values \mathcal{C} obtained for the RTISI-LA algorithm are usually better than the ones obtained with Griffin and Lim, but only in the order of 6dB of improvement. This improvement is mainly based on the emphasis on time coherence of the signal, as construction is done in both ways (forward and backward). Additional work from Gnann et al. [27] has focused on the phase initialization and processing order of the reconstruction. By processing the frame according to their energy and initializing the phase with unwrapping, one can improve the convergence of the reconstruction by 1 to 5dB.

Additional work from Le Roux [22] showed the same tendency when adding the phase initialization of RTISI-LA to the STFT consistency-based reconstruction.

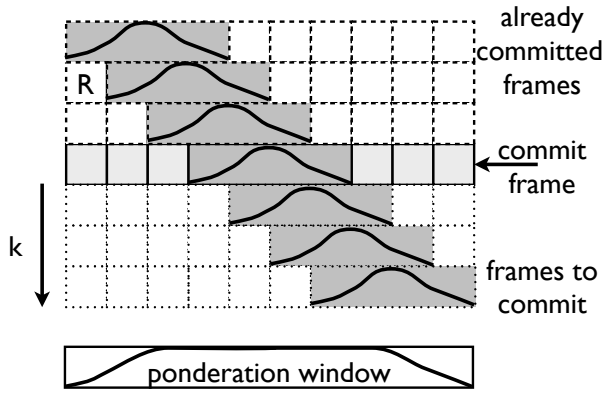


Figure 8: Real Time Iterative Spectrogram Inversion with Look Ahead of $k = 3$ and 75% of overlap.

6.3. Summary on existing techniques

Existing techniques are gradually introducing more and more constraints in the time domain, compared to the first approach of Griffin and Lim. They are still providing results that are close to the original spectrogram (convergence in the \mathcal{C} criterium) but far from the original time domain signal. This tendency to generate incoherent signals in the time domain will be explained in the section 8 addressing fundamental issues shared by these current approaches.

Informal experiments were done using the initialization proposed in condition 5 and 6 of section 3 (knowledge of first samples of the signal) using the RTISI-LA technique. Unfortunately, this condition was not able to improve the reconstruction quality. Indeed, these conditions are neither necessary nor sufficient to perform perfect signal reconstruction, with both STFT coherence or real time spectrogram inversion.

7. INJECTING ADDITIONNAL INFORMATIONS

The three algorithms presented before do not show high accuracy in the reconstruction of the signal. Reconstruction errors \mathcal{R} are often above zero, and rarely below $-6dB$. Therefore, injecting additional information on the signal could be a possible way to achieve a better reconstruction.

As perfect signal reconstruction involves very small variations on the spectrograms, much lower than the convergence values \mathcal{C} usually obtained with the previous methods, one solution is to inject additional information during reconstruction. This information can be a prior on the shape of the signal, local phase information or shape criterion.

7.1. Additional knowledge on the signal spectrum

Alsteris and al. [5] have proposed an extended study on the possibility of reconstructing the signal while only knowing partial information of the spectrum and especially the knowledge of phase sign, phase delay or group delay. Moreover, when a prior on the position of the poles and zeros of the z-transform of each frame of the STFT is known, reconstruction can be made using the known relations between amplitude and phase of a DFT [7].

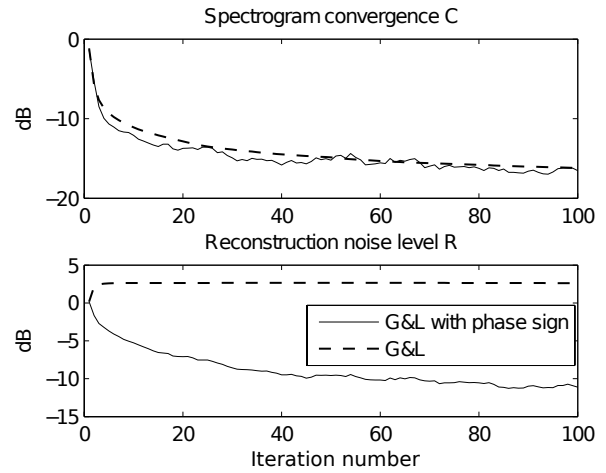


Figure 9: Convergence and reconstruction noise level for Griffin and Lim's method, with and without knowledge of the sign of the original phase.

Phase sign, alternatively, has been shown to be a powerful addition to the spectrogram [28] in order to achieve a reconstruction of good quality for a very small amount of extra information (only one bit per bin). However, such information is not always available, especially in the case of blind source separation when the signal to be reconstructed is not known well enough. New approaches such as informed source separation could however benefit from the information of phase sign.

On figure 9, both convergence \mathcal{C} and reconstruction noise \mathcal{R} are shown for the Griffin and Lim reconstruction (512 samples half sinus window with 75% overlap) with or without knowledge of the phase sign. The test signal is a music sample of 2 seconds, sampled at 44.1kHz. One can see that phase sign does not improve the convergence speed of the algorithm in terms of \mathcal{C} , but dramatically enhances the quality of the reconstruction, as \mathcal{C} and \mathcal{R} become strongly correlated. Perceptively, transients are better reconstructed with less smearing and artifacts.

However, as shown with this example, sign information does not seem sufficient to achieve perfect reconstruction in practice, as the reconstruction noise levels \mathcal{R} remain high even after 100 iterations. However, convergence could probably be faster while using this prior on phase sign for proper initialization of the algorithm.

7.2. Probabilistic inference

Another idea that has been explored is to use some statistical properties of the signal. The work proposed by Achan [29] uses an autoregressive model of the speech signal to be reconstructed, in order to improve the convergence of the algorithm. As mentioned in the article, the proposed method performs only slightly better than the classic Griffin and Lim (approx. 2 to 4dB depending on the model) and resorts to a posteriori regularization of the signal. This can however be an interesting approach when the class of signals to be recovered is well defined. Also, the idea beneath this technique is interesting, as concurrent optimization is done both in the time and STFT domain, whereas blind techniques only constrain the STFT domain.

7.3. Local observations

Spectrograms also possess local properties that can be extracted with or without a prior in order to recover the original signal:

Nouvel [30] proposed the iterative estimation of local patterns of the time-frequency diagram, patterns based on a polynomial expression of the phase, for instance. The algorithm proposed performs better than Griffin and Lim only when there is no overlap. Missing information is then brought to the reconstruction by the prior learning of the polynomial coefficients.

Another approach is the Max-Gabor analysis of spectrograms from Ezzat et al. [31]. It uses local patch of the spectrogram where local amplitude, frequency, orientation and phases are estimated. The information are used in order to synthesize the time-domain signal with Gabor functions. Unfortunately this study does not address the quality of the reconstruction by comparing it to Griffin and Lim as it was not aimed originally at the task of phase recovery.

7.4. Conclusion: usefulness of additional information

In this section we presented some recent techniques that perform signal reconstruction from spectrogram while having additional informations on the signal to reconstruct. We saw that despite some advanced models, the proposed algorithms are only slightly better than the original framework from Griffin and Lim, especially in terms of the time-domain \mathcal{R} error criterium.

Even when using the sign of the STFTs, Griffin and Lim algorithm does not convergence faster, nor better: only the quality (SNR) improves. This proves that most of the work to improve the convergence has to be done on the reconstruction algorithms themselves, as additional information only serves at improving the final quality. The issues that are preventing the convergence despite the additional information are discussed in the next section.

8. OVERLOOKED ISSUES

As far as the state of the art goes, a number of issues regarding signal reconstruction from spectrograms seem overlooked. One of them is the use of the convergence criterion \mathcal{C} which requires extremely high convergence (difference of approx. -90dB) in order to achieve a perfect reconstruction of the signal. Other issues are caused by the way information is spread in the spectrogram or by the minimization technique of the reconstruction itself.

8.1. Phase information and spectrogram

The first major issue of signal reconstruction from spectrograms is the effect of phase information in the modulus of the STFT. Because the STFT is obtained via windowing, one can find at bin n the contribution of many spectral components added to one another, thus forming a linear system [13, 14, 32]. However, such system only finds a suitable solution under three precautions:

1. The analysis window has to produce a lot of spectral leakage. The Gaussian window is a good example of such window and is often used.
2. The overlap has to be very high, in order to provide as little time downsampling as possible in every frequency channel.
3. Usually DFT are oversampled, bringing yet another layer of redundancy in the STFT

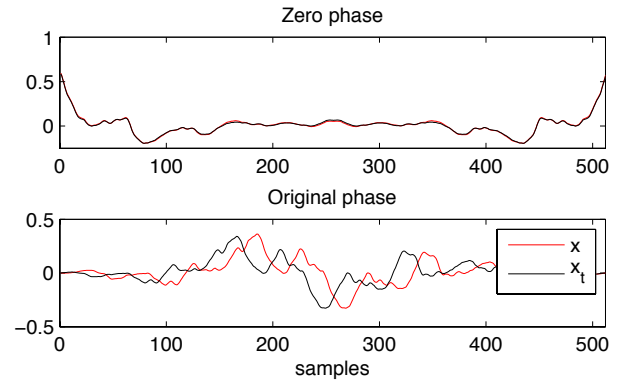


Figure 10: Spectrogram amplitude difference with or without phase for two signal x and x_t . x_t is the signal x translated 20 samples to the left. Spectral difference \mathcal{C} between the two frames is -25dB.

In real analysis conditions, when using windows with a low spectral leakage and a rather low overlap (usually 50% or 75%), such an analytic resolution of the system is not possible, mainly due to the precision of both the data contained in the STFT and the complexity of the system to solve.

One example is given on figure 10 where the same frame of two different STFTs of a speech signal sampled at 16kHz and quantized on 16 bits are displayed: in red, the frame inverse DFT of a frame of the STFT of original signal x and in black the same frame of the STFT of x_t , the signal x shifted by 20 samples to the left. On the top row, the inverse DFTs are presented with zero phase (magnitude only) and on the bottom row the time-domain inverse DFTs with the original phase information are given. Despite the vast difference between the two frames, the zero phase responses are very similar (differences are barely visible around samples 160 and 350). Difference \mathcal{C} of the two signals on the top row of figure 10 is -25dB, approx. the convergence limit of Griffin and Lim's technique. Although this figure is a good example of the poor effect of phase on the magnitude of the STFT, it will also serve well the illustration of stagnation by translation given later.

8.2. Stagnation caused by sign indetermination

Fienup et al. [17] proposed an interesting study on the problems preventing iterative algorithms such as Griffin and Lim's to converge toward a unique solution. It described this issues as *stagnation*, a self explanatory term that illustrates the inability of the algorithm to converge toward an optimal solution because it reached a local minima of optimization. Although Fienup's work was based on image processing, two of the three stagnations described in [17] can very well be observed on one-dimensional signals.

The first stagnation is linked to the sign indetermination illustrated in section 3. During reconstruction, the algorithm can be stuck between a mix of the two possible solutions x and $-x$, because it converged toward features of both signals. This phenomenon is illustrated on figure 11. On this figure, one can see that at the beginning the estimated signal \tilde{x} is in phase with x whereas at the end it is in phase with $-x$. On the middle on the figure, one can see a characteristic point when \tilde{x} gets closer to zero, illustrating an inflection point from one frame to another. Note that the

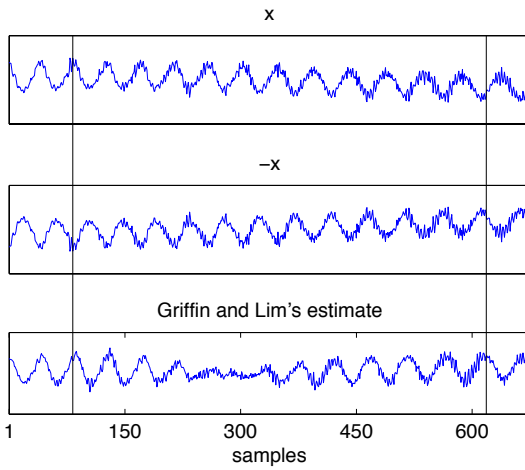


Figure 11: The first stagnation: the algorithm estimation (bottom) is stuck between a mix of x (top) and $-x$ (middle). Estimation with an half sinus window of 512 sample long, overlap of 75%.

difference between the two local minima is approximately equal to the window size. Such stagnation is also observed on signals reconstructed with RTISI-LA and the STFT coherence. Moreover, this stagnation is not consistent along the frequency axis: a closer look to the signal presented on figure 11 shows that phase coherence toward x or $-x$ is only true for the first harmonic.

This first stagnation is countered by the knowledge of the sign of the STFT presented in section 7 and is the main cause of the very high noise estimation levels \mathcal{R} observed when reconstructing a signal with either of the three method presented in section 6. Basically, knowing the sign of the STFT causes the uniqueness of the solution to be true, avoiding a lot of local minima during minimization.

An other stagnation, also explained by the sign indetermination is what Fienup called "fringes". Sadly, this observation is hard to make on audio signals but is still present during the reconstruction. Because of this sign indetermination $|DFT[x(-n)]| = |DFT[x(n)]|$, frames happen to be estimated in the wrong time direction. Most of the times, overlap is enough to prevent such stagnation, which is then the most unlikely to happen.

Solutions to overcome these stagnations proposed in [17] do not apply well to signal processing, as they were designed for image processing. However, the idea of Monte Carlo method and artificial boundaries of the reconstruction seem interesting and easily transposable to the signal domain.

8.3. Stagnation caused by translation

The third stagnation is the translation of the signal. Because the TFD operator is circular, translation of the signal does not always drastically change the magnitude of the transform (figure 10) despite the windowing. Therefore, convergence can happen to a translated version of the original signal: like in figure 12 where a signal and its reconstruction with Griffin and Lim's technique are presented. This problem can be linked to the phase rotation problem addressed in section 3 but on local portions of the signal.

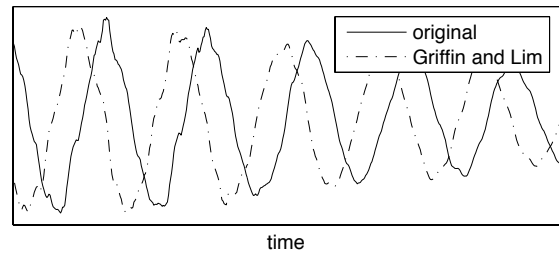


Figure 12: Stagnation by translation for a Griffin and Lim reconstruction (half sinus 512 sample window, 75% overlap, 200 iterations)

8.4. Different stagnation per frequency band

An other issue of the stagnation is that it happens at different levels on different frequency bands. Because the coherence of the STFT is limited (surfaces of figure 6) in both time and frequency, a gap in energy can cause different patches of the reconstructed STFT to present different kinds of stagnation. As music signal often presents harmonic structure or colored noise, localized energy is very common.

An illustration of this phenomenon is given on figure 13 where a speech signal (the original, 16bits and 16kHz, at approx. 200Hz fundamental) and its reconstruction from its spectrogram (Griffin and Lim, 512 sample half sinus window, 200 iterations) are showed for different frequency band. The filter bank presents a passing band of 400Hz and a zero phase to prevent delay to be inserted between observations.

On the two first bands, from 1600 to 4600Hz, the signal is well reconstructed and is mainly presenting a small stagnation by translation. However, the direction of the translation is not the same for the two bands.

On the bands three to five, one can mainly see a stagnation by sign indetermination with characteristic inflection points based around samples 2300 and 2460 for band 3, 2375 for band 4 and 2325, 2495 for band 5. Once again, even if the bands are presenting the same type of stagnation, their evolution is different, mainly dependent on the local frequency.

It can be noted that, as expected, this stagnation issue gets more and more problematic as frequency increases. At low frequencies, the overlap between adjacent windows represents a smaller phase increment than at high frequencies. This may give an insight on why standard phase reconstruction offers a rather good sound quality despite a low SNR: at high frequencies, the ear is not so sensitive to the phase but rather to the general energy in the frequency bands. It may also indicate that algorithms based in the injection of additional information should have different trade-offs in terms of precision versus amount of extra information, in different frequency bands.

9. APPLICATIONS TO DIGITAL AUDIO PROCESSING

In the case of source separation in a linear instantaneous stationary mixture, one often knows partial information on the source to be reconstructed, such as its spectrogram (or corrupted spectrogram). In this case, Gunawan [9] proposed a framework in order to use the information contained in the mixture M_x of N sources to help the

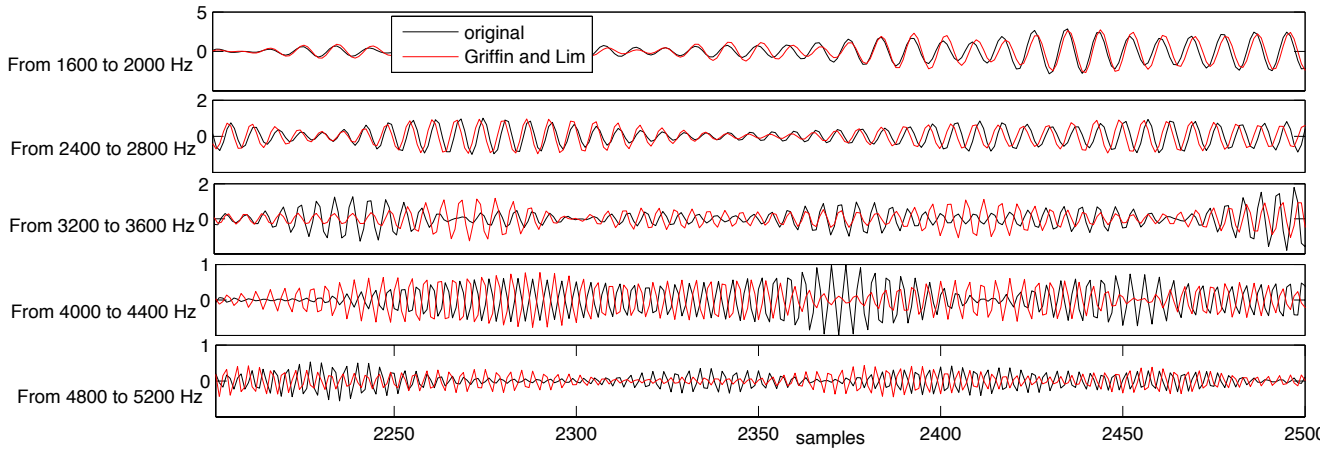


Figure 13: Signal Comparison (original in black, Griffin and Lim's reconstruction in red) for different frequency bands (zero phase filter bank). Stagnation are not consistent across frequency

phase estimation. While constraining the spectrogram W_j of the j -th source, one can reconstruct its phase with the following steps:

$$\hat{S}_j^{k+1} = \sqrt{W_j} e^{i\angle STFT(STFT^{-1}(S_j^k + \frac{e^k}{N}))} \quad (11)$$

$$e^{k+1} = M_x - \sum_j \hat{S}_j^{k+1} \quad (12)$$

This way, stagnations such as sign indetermination or translation are automatically compensated by the error computed on equation (12). The phase of the mixture is used as an additional information to constrain the reconstruction. Of course, this study provides the best results when the target spectrogram of each source W_j is perfectly known, while in practice the target spectrogram is only an estimate. Results are also conditioned by the number N of sources, with the best results for only 2 sources.

An other study [10] proposed by Le Roux used the spectrogram consistency (the fact that $S = STFT(STFT^{-1}S)$) as a constraint for the maximum likelihood estimation of a Wiener filter α_j for the j -th source. Such filters are used to perform adaptive filtering (for instance, in denoising) but usually rely on the energy ratio between the sources:

$$\alpha_j(n, m) = \frac{W_j(n, m)}{\sum_k W_k(n, m)} \quad (13)$$

$$\hat{S} = \alpha M \quad (14)$$

By explicitly adding the constraint that

$$\hat{S}_j - STFT(STFT^{-1}\hat{S}_j) = 0$$

into the equation, results show an improvement in SNR of around 3dB when applied on speech denoising.

10. SUMMARY AND CONCLUSION

In this paper we presented a state of the art on the question of signal reconstruction from spectrogram. We especially addressed the problem of perfect reconstruction and the issues preventing existing algorithms from converging to one (or one of the possible) solution.

Unicity is an important question to be asked in this case, but ordinary conditions are sufficient to guarantee that there is no more than two possible solutions for the reconstruction, given by the sign indetermination of the magnitude operator. Still, we saw that duplicity of the solution is the cause of the stagnation of the minimization by sign indetermination.

The three current techniques of blind reconstruction (Griffin and Lim, RTISI-LA and STFT coherence) have been described and discussed. Although there has been more than 20 years between Griffin and Lim's and the two other techniques, overall reconstruction quality has not significantly improved. Of course, computation time and implementation (especially in the case of real-time processing) have been a huge development part of such techniques, but we feel that most of the work has yet to be focused on the actual process leading to the optimal convergence of the algorithm in order to get better than just perceptively close reconstructions.

Given the amount of information present in the spectrogram, especially with the typical value of 75% overlap, perfect reconstruction (i.e. reconstructing x from $|STFT[x]|$ with error inferior to the measure error on x itself) should be possible. We raised however a number of issues preventing convergence of the reconstruction toward the absolute minima. Those issues, called stagnation by Fienup [17] are configurations that prevent further minimization of the error. Stagnation presented are of two types: stagnation by sign indetermination (time inversion and signal inversion) and stagnation by translation. Because music signals are not evenly distributed on the time-frequency plan, stagnation can occur independently on local patches of the spectrogram both in time and frequency and is therefore difficult to correct.

Future work should then emphasize the resolution of the stagnation problems highlighted in this article, either with side information or using blind reconstruction. Whereas solving the problem of sign indetermination should be rather simple as one can observe sign coherent patches in the reconstructed STFT, phase translation is more problematic as it produces time delay that varies for the whole time-frequency domain.

11. REFERENCES

- [1] L. Benaroya, F. Bimbot, and R. Gribonval, "Audio source separation with a single sensor," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 191–199, 2006.
- [2] Guoshen Yu, S. Mallat, and E. Bacry, "Audio denoising by time-frequency block thresholding," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1830–1839, may 2008.
- [3] D.W. Griffin and J.S. Lim, "Signal reconstruction from short-time fourier transform magnitude," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 32(2), pp. 236–243, 1984.
- [4] M. Portnoff, "Implementation of the digital phase vocoder using the fast fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 24, no. 3, pp. 243–248, June 1976.
- [5] Leigh D. Alsteris and Kuldip K. Paliwal, "Iterative reconstruction of speech from short-time fourier transform phase and magnitude spectra," *Computer Speech & Language*, vol. 21, no. 1, pp. 174–186, 2007.
- [6] K.K. Paliwal and L.D. Alsteris, "On the usefulness of stft phase spectrum in human listening tests," *Speech Communication*, vol. 45 (2), pp. 153–170, 2005.
- [7] B. Yegnanarayana, D. Saikia, and T. Krishnan, "Significance of group delay functions in signal reconstruction from spectral magnitude or phase," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, no. 3, pp. 610–623, June 1984.
- [8] Jonathan Le Roux, Hirokazu Kameoka, Nobutaka Ono, and Shigeki Sagayama, "Fast signal reconstruction from magnitude stft spectrogram based on spectrogram consistency," *proceedings of DAFx'10*, 2010.
- [9] D. Gunawan and D. Sen, "Iterative phase estimation for the synthesis of separated sources from single-channel mixtures," *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 421–424, may 2010.
- [10] Jonathan Le Roux, Emmanuel Vincent, Yuu Mizuno, Hirokazu Kameoka, Nobutaka Ono, and Shigeki Sagayama, "Consistent Wiener filtering: Generalized time-frequency masking respecting spectrogram consistency," in *Proc. 9th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA 2010)*, Sept. 2010, pp. 89–96.
- [11] Bin Yang, "A study of inverse short-time fourier transform," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008*, 4 2008, pp. 3541–3544.
- [12] S. Nawab, T. Quatieri, and Jae Lim, "Signal reconstruction from short-time fourier transform magnitude," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, no. 4, pp. 986–998, Aug. 1983.
- [13] Jake Bouvrie and Tony Ezzat, "An incremental algorithm for signal reconstruction from short-time fourier transform magnitude," *proceedings of ICSLP'06*, 2006.
- [14] Radu Balan, "On signal reconstruction from its spectrogram," *proceedings of the Conference on Information and Sciences Systems*, 2010.
- [15] M. Hayes, Jae Lim, and A. Oppenheim, "Signal reconstruction from phase or magnitude," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, pp. 672–680, 1980.
- [16] W. Kim and M.H. Hayes, "Phase retrieval using a window function," *IEEE Transactions on Signal Processing*, vol. 41, no. 3, pp. 1409–1412, Mar. 1993.
- [17] J. R. Fienup and C. C. Wackerman, "Phase-retrieval stagnation problems and solutions," *J. Opt. Soc. Am. A*, vol. 3, pp. 1897–1907, 1986.
- [18] Jonathan Le Roux, Nobutaka Ono, and Shigeki Sagayama, "Explicit consistency constraints for stft spectrograms and their application to phase reconstruction," *proceedings of SAPA'08*, 2008.
- [19] M. Hayes and T. Quatieri, "The importance of boundary conditions in the phase retrieval problem," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '82.*, May 1982, vol. 7, pp. 1545–1548.
- [20] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of the phase from image and diffraction plane pictures," *Optik*, vol. 35, pp. 237–246, 1972.
- [21] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *proc. of the IEEE*, pp. 51–83, 1978.
- [22] Jonathan Le Roux, Hirokazu Kameoka, Nobutaka Ono, and Shigeki Sagayama, "Phase initialization schemes for faster spectrogram-consistency-based signal reconstruction," in *Proceedings of the Acoustical Society of Japan Autumn Meeting*, Mar. 2010, number 3-10-3.
- [23] Xinglei Zhu, G. Beauregard, and L. Wyse, "Real-time signal estimation from modified short-time fourier transform magnitude spectra," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 5, pp. 1645–1653, 2007.
- [24] G. T. Beauregard, X. Zhu, and L. Wyse, "An efficient algorithm for real-time spectrogram inversion," in *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05)*, Sept. 2005, pp. 116–118.
- [25] Volker Gnann and Martin Spiertz, "Inversion of short-time fourier transform magnitude spectrograms with adaptive window lengths," *proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 325–328, 2009.
- [26] X. Zhu, G. Beauregard, and L. Wyse, "Real-time iterative spectrum inversion with look-ahead," *proceedings of IEEE International Conference on Multimedia and Expo*, pp. 229–232, 2006.
- [27] Volker Gnann and Martin Spiertz, "Improving rtisi phase estimation with energy order and phase unwrapping," *proceedings of DAFx'10, Gratz, Austria*, 2010.
- [28] P. Van Hove, M. Hayes, Jae Lim, and A. Oppenheim, "Signal reconstruction from signed fourier transform magnitude," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, no. 5, pp. 1286–1293, Oct. 1983.
- [29] Kannan Achan, Sam T. Roweis, and Brendan J. Frey, "Probabilistic inference of speech signals from phaseless spectrograms," in *In Neural Information Processing Systems 16*, 2003, pp. 1393–1400, MIT Press.

- [30] Bertrand Nouvel, “A study of a local-features-aware model for the problem of phase reconstruction from the magnitude spectrogram,” *proceedings of ICASSP'10*, pp. 4026–4029, 2010.
- [31] Tony Ezzat, Jake Bouvrie, and Tomaso Poggio, “Max-gabor analysis and synthesis of spectrograms,” *proceedings of IC-SLP'06*, 2006.
- [32] M.R. Portnoff, “Magnitude-phase relationships for short-time fourier transforms based on gaussian analysis windows,” *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 79*, vol. 4, pp. 186–189, 1979.

MODAL ANALYSIS OF IMPACT SOUNDS WITH ESPRIT IN GABOR TRANSFORMS

A. Sirdey, O. Derrien, R. Kronland-Martinet, *

Laboratoire de Mécanique et d'Acoustique
CNRS

Marseille, France

<name>@lma.cnrs-mrs.fr

M. Aramaki, *

Institut de Neurosciences cognitives de la Méditerranée
CNRS

Marseille, France

<name>@incm.cnrs-mrs.fr

ABSTRACT

Identifying the acoustical modes of a resonant object can be achieved by expanding a recorded impact sound in a sum of damped sinusoids. High-resolution methods, e.g. the ESPRIT algorithm, can be used, but the time-length of the signal often requires a sub-band decomposition. This ensures, thanks to sub-sampling, that the signal is analysed over a significant duration so that the damping coefficient of each mode is estimated properly, and that no frequency band is neglected. In this article, we show that the ESPRIT algorithm can be efficiently applied in a Gabor transform (similar to a sub-sampled short-time Fourier transform). The combined use of a time-frequency transform and a high-resolution analysis allows selective and sharp analysis over selected areas of the time-frequency plane. Finally, we show that this method produces high-quality re-synthesized impact sounds which are perceptually very close to the original sounds.

1. INTRODUCTION

The context of this study is the identification of acoustical modes which characterize a resonant object, which is of great use when building an environmental sound synthesizer (see [1] or [2] for an insight on such synthesizers). Practically, the analysis is made from recorded impact sounds, where the resonant object is hit by another solid object (e.g. a hammer). Assuming that the impact sound is approximately the acoustical impulse response of the resonant object, each mode corresponds to an exponentially damped sinusoid (EDS). The modal analysis thus consists of estimating the parameters of each sinusoidal component (amplitude, phase, frequency and damping). These parameters will be stored, and eventually modified, before further re-synthesis. In this paper, we consider only the analysis part.

In the past decades, significant advances have been made in the field of system identification, especially for estimating EDS parameters in a background noise. Although the so-called *high-resolution methods* or *subspace methods* (MUSIC, ESPRIT) [3, 4] were proved to be more efficient than spectral peak-picking and iterative analysis-by-synthesis methods [5], few applications have been proposed. One can suppose that the high computational complexity of these methods is a major drawback to their wide use: on a standard modern computer, the ESPRIT algorithm can hardly analyse more than 10^4 samples, which corresponds roughly to 200 ms sampled at 44100 Hz. This is usually too short for analysing

properly impact sounds which can last up to 10 s. Sub-band decomposition with critical sub-sampling in each band seems to be a natural solution to overcome the complexity problem, as it has already been shown in [6] and [7]. Another drawback is that ESPRIT gives accurate estimates when the background noise is white, which is usually not the case in practical situations. This problem can be overcome by the use of whitening filters. The estimation of the model order (i.e. the number of modes) is also an important issue. Various methods have been proposed for automatic estimation of the order, e.g. ESTER [8], but this parameter is often deliberately over-estimated in most practical situation.

In this paper, we propose a novel method for estimating the modes with ESPRIT algorithm: we first apply a Gabor Transform (GT), which is basically a sub-sampled version of the short-time Discrete Fourier Transform (DFT), to the original sound in order to perform a sub-band decomposition. The number of channels and the sub-sampling factor depend on the Gabor frame associated to the transform. We show that an EDS in the original sound is still an EDS inside each band, and the original parameters can be recovered from a sub-band analysis using ESPRIT. Furthermore, if the number of frequency sub-bands is high enough, it is reasonable to assume that the noise is white inside each sub-band. We also propose a method to discard insignificant modes a posteriori in each sub-band.

The paper is organised as follows: first, in a brief state-of-the-art, we describe the signal model, the ESPRIT algorithm and the Gabor transform. Then, we show that original EDS parameters can be recovered by applying the ESPRIT algorithm in each frequency band of the Gabor transform. In the next part, we describe an experimentation on a real metal sound, and show the efficiency of our method. Finally, we discuss further improvements.

2. STATE OF THE ART

2.1. The signal model and the ESPRIT algorithm

The discrete signal to be analysed is written:

$$x[l] = s[l] + w[l] \quad (1)$$

where the deterministic part $s[l]$ is a sum of K damped sinusoids:

$$s[l] = \sum_{k=0}^{K-1} \alpha_k z_k^l \quad (2)$$

where the complex amplitudes are defined as $\alpha_k = a_k e^{i\phi_k}$ (containing the initial amplitude a_k and the phase ϕ_k), and the poles are defined as $z_k = e^{-d_k + 2i\pi\nu_k}$ (containing the damping d_k and

* The research leading to this paper was supported by the French GIP ANR under contract ANR-00301, Métaphores Sonores - METASON. See the project website <http://metason.cnrs-mrs.fr/> for more details.

the frequency ν_k). The stochastic part $w[l]$ is a gaussian white noise of variance σ^2 .

The ESPRIT algorithm was originally described by Roy *et. al.* [4], but many improvements have been proposed. Here, we use the Total Least Square method by Van Huffel *et. al* [9]. The principle consists of performing a SVD on an estimate of the signal correlation matrix. The eigenvectors corresponding to the K highest eigenvalues correspond to the so called *signal space*, while the remaining vectors correspond to the so called *noise space*. The shift invariance property of the signal space allows a simple solution for the optimal poles values z_k . Then, the amplitudes α_k can be recovered by solving a least square problem. The algorithm can be described briefly as follows:

We define the signal vector:

$$\mathbf{x} = [x[0] \ x[1] \ \dots \ x[L-1]]^T, \quad (3)$$

where L is the length of the signal to be analysed. The Hankel signal matrix is defined as:

$$\mathbf{X} = \begin{bmatrix} x[0] & x[1] & \dots & x[Q-1] \\ x[1] & x[2] & \dots & x[Q] \\ \vdots & \vdots & & \vdots \\ x[R-1] & x[R] & \dots & x[L-1] \end{bmatrix} \quad (4)$$

where $Q, R > K$ and $Q + R - 1 = L$. We also define the amplitude vector:

$$\boldsymbol{\alpha} = [\alpha_0 \ \alpha_1 \ \dots \ \alpha_{K-1}]^T, \quad (5)$$

and the Vandermonde matrix of the poles:

$$\mathbf{Z}^L = \begin{bmatrix} 1 & 1 & \dots & 1 \\ z_0 & z_1 & \dots & z_{K-1} \\ \vdots & \vdots & & \vdots \\ z_0^{L-1} & z_1^{L-1} & \dots & z_{K-1}^{L-1} \end{bmatrix}. \quad (6)$$

Performing a SVD on \mathbf{X} leads to:

$$\mathbf{X} = [\mathbf{U}_1 \mathbf{U}_2] \begin{bmatrix} \boldsymbol{\Sigma}_1 & 0 \\ 0 & \boldsymbol{\Sigma}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \end{bmatrix}, \quad (7)$$

where $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are diagonal matrix containing respectively the K largest singular values, and the smallest singular values; $[\mathbf{U}_1 \mathbf{U}_2]$ and $[\mathbf{V}_1 \mathbf{V}_2]$ are respectively the corresponding left and right singular vectors. The shift-invariance property of the signal space leads to:

$$\mathbf{U}_1^\perp \boldsymbol{\Phi}_1 = \mathbf{U}_1^\perp, \quad \mathbf{V}_1^\perp \boldsymbol{\Phi}_2 = \mathbf{V}_1^\perp, \quad (8)$$

where the eigenvalues of $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ provide an estimation of the poles z_k . $(\cdot)^\dagger$ and $(\cdot)^\downarrow$ respectively stand for the operators discarding the first line and the last line of a matrix. Thus, z_k can be estimated by diagonalization of matrix $\boldsymbol{\Phi}_1$ or $\boldsymbol{\Phi}_2$. The associated Vandermonde matrix \mathbf{Z}^L is computed. Finally, the optimal amplitudes with respect to the least square criterion are obtained by:

$$\boldsymbol{\alpha} = (\mathbf{Z}^L)^\dagger \mathbf{x}, \quad (9)$$

where $(\cdot)^\dagger$ denotes the pseudoinverse operator.

2.2. The Gabor Transform

The Gabor transform of signal $x[l]$ can be written as:

$$\chi[m, n] = \sum_{l=0}^{L-1} \bar{g}[l - an] x[l] e^{-2i\pi l \frac{m}{M}}, \quad (10)$$

where $g[l]$ is the analysis window, a is the time-step and M the number of frequency channels. (\cdot) denotes the complex conjugate. m is a discrete frequency index and n a discrete time-index. $\{g, a, M\}$ is called a *Gabor frame*. For some frames, this transform can be inverted. A necessary condition is $a \leq M$ (for more details, see for instance [10]). The signal $\chi[m, n]$ for a fixed index m can be seen as a sub-sampled and band-pass filtered version of the signal $x[l]$. As the sub-sampling reduces the length of the data, we apply the ESPRIT algorithm to each frequency channel in order to analyse longer signals.

3. ESPRIT IN A GABOR TRANSFORM

In this section, we investigate the application of the ESPRIT algorithm to a single channel of the GT. As the GT is linear, we separate the contribution of the deterministic part $s[l]$ and the contribution of the noise $w[l]$.

3.1. Deterministic part

We denote $c[m, n]$ the GT of $s[l]$ in channel m and time index n . We also note $c_k[m, n]$ the GT of the signal z_k^l associated to the pole z_k :

$$c_k[m, n] = \sum_{l=0}^{L-1} \bar{g}[l - an] z_k^l e^{-2i\pi l \frac{m}{M}}. \quad (11)$$

According to the signal model (2), it can be easily proved that:

$$c[m, n] = \sum_{k=0}^{K-1} \tilde{\alpha}_{k,m} \tilde{z}_{k,m}^n, \quad (12)$$

where the apparent pole $\tilde{z}_{k,m}$ can be written as:

$$\tilde{z}_{k,m} = z_k^a e^{-2i\pi a \frac{m}{M}}, \quad (13)$$

and the apparent amplitude:

$$\tilde{\alpha}_{k,m} = \alpha_k c_k[m, 0]. \quad (14)$$

In other words, the deterministic part of the signal in each channel is still a sum of exponentially damped sinusoids, but the apparent amplitudes and phases are modified.

3.2. Stochastic part

Assuming that the time-step a is close to M ensures that the GT of the noise in each channel is approximately white. Furthermore, it has been proved that the Gabor transform of a gaussian noise is a complex gaussian noise [11]. So we assume that the GT of $w[l]$ in each channel is a complex white gaussian noise.

3.3. Recovering the signal parameters

As the signal model is still valid, it is reasonable to apply ESPRIT on $c[m, n]$. We note \mathbf{c}_m the vector of GT coefficients in the channel m and \mathbf{S}_m the Hankel matrix built from $c[m, n]$. Applying the ESPRIT algorithm to \mathbf{S}_m leads to the estimation of the apparent poles $\tilde{z}_{k,m}$. Inverting equation (13) leads to:

$$z_k = e^{2i\pi \frac{m}{M}} (\tilde{z}_{k,m})^{\frac{1}{a}}. \quad (15)$$

Because of the sub-sampling introduced by the GT, it can be seen from equation (13) that aliasing will occur when the frequency of a pole is outside the interval $[\frac{m}{M} - \frac{1}{2a}, \frac{m}{M} + \frac{1}{2a}]$. To avoid aliasing, we choose the analysis window $g[l]$ so that its bandwidth is smaller than $\frac{1}{a}$. That way, the possible aliasing components will be attenuated by the band-pass effect of the Gabor transform.

We note $\tilde{\mathbf{Z}}_m^N$ the Vandermonde matrix of the apparent poles $\tilde{z}_{k,m}$ (N is the time-length of signal $c[m, n]$). The least square method for estimating the amplitudes leads to:

$$\alpha = \frac{(\tilde{\mathbf{Z}}_m^N)^\dagger \mathbf{c}_m}{c_k[m, 0]}. \quad (16)$$

Without noise, according to equation (12), each EDS should be detected in each channel, which generates multiple estimations of the same modes. Theoretically, the model order should be set to K in each channel. However, this is usually a large over-estimation. Because each channel of the GT behaves like a band-pass filter, an EDS with a frequency far from $\frac{m}{M}$ will be attenuated and considered as noise. Thus practically, the exact number of detectable components in each channel is unknown. So we set the model order in each channel with the ESTER criterion (see section 4.3 for implementation details).

4. EXPERIMENTATION

When applied on synthetical sounds that strictly verify the signal model (1), the full-band ESPRIT algorithm, as well as the ESTER criteria, estimate the model parameters with an excellent precision (see [6], [8]). Estimation errors are observed when dealing with real-life sounds. Therefore this section does not consider the analysis of synthetical sounds, but focuses on the analysis/synthesis of a real metal sound *m5* (which can be listened to at [12]). *m5* has been produced hitting a metal plate with a drum stick. Observing its waveform, Fourier transform and spectrogram (Fig. 4a, 4e and 1) one can see that it presents a rich spectral content and significant lasting energy up to 6 s.

4.1. Analysis with full-band ESPRIT method

Considering the size of the Hankel matrix corresponding the whole sound (around 150000×150000), only a part of the original signal can be analysed with the full-band ESPRIT algorithm. Fig. 2 shows the ESTER criteria cost function computed for the 10000 first samples of *m5*. The optimal model order theoretically corresponds to the maximum of this function, which is reached here for $K = 4$ modes. This value is obviously not consistent, as one can see on the spectrogram of *m5*: the spectral content is obviously much more complex. A reasonable compromise would be to choose the maximum order for which the cost function is above a given threshold. For instance, this threshold can be set to 100. The corresponding model order is $K = 206$. After applying the

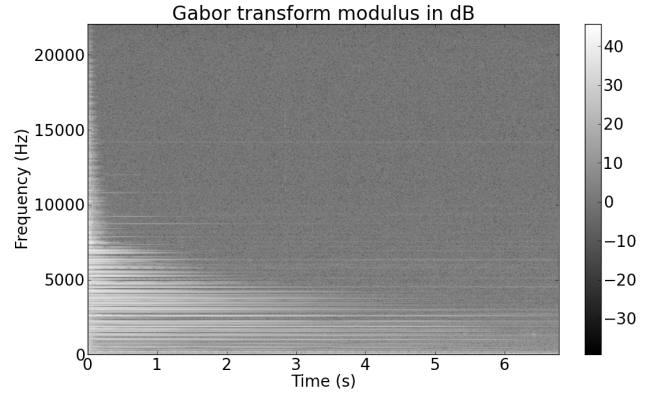


Figure 1: Spectrogram of *m5*.

ESPRIT algorithm, 29 EDS appear to have a negative damping, which will form diverging components at the re-synthesis. Since they do not describe physical modes, they must be discarded. The resulting synthesised sound *m5_std_esprit* ([12]) is unsatisfying from a perceptual point of view, and reveals that the damping behaviour of some modes has been wrongly estimated as well. Furthermore there is a significant difference in the spectral content of the original and the re-synthesized sound above 12000 Hz, as shown by Fig. 3 and 4e.

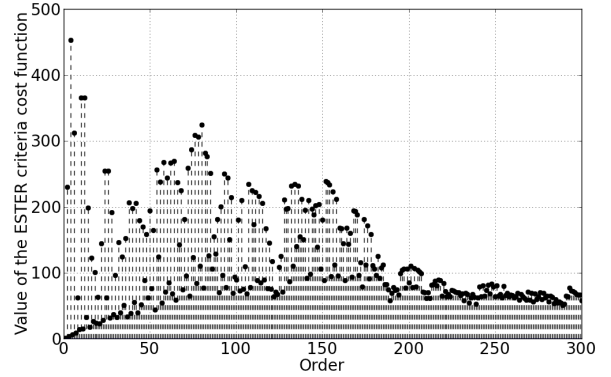


Figure 2: ESTER criteria cost function computed for the 10000 first samples of the full-band signal *m5*.

4.2. Analysis with ESPRIT in a Gabor transform

The chosen Gabor frame consists in a Blackman-Harris window of length 1024, a time-step parameter $a = 32$, and a number of channels $M = 1024$. It is unnecessary to apply the ESPRIT algorithm over regions of the time-frequency plane that only contain noise. Since the most important deterministic information is contained in the channels of high energy, those channels can be identified using a peak detection algorithm over the energy profile of the Gabor transform as shown in Fig. 5. In a software environment, the choice of which channels will be analysed could be left to the user. It is reasonable to think that the noise whitening induced by the sub-band division of the spectrum makes the ESTER criteria more

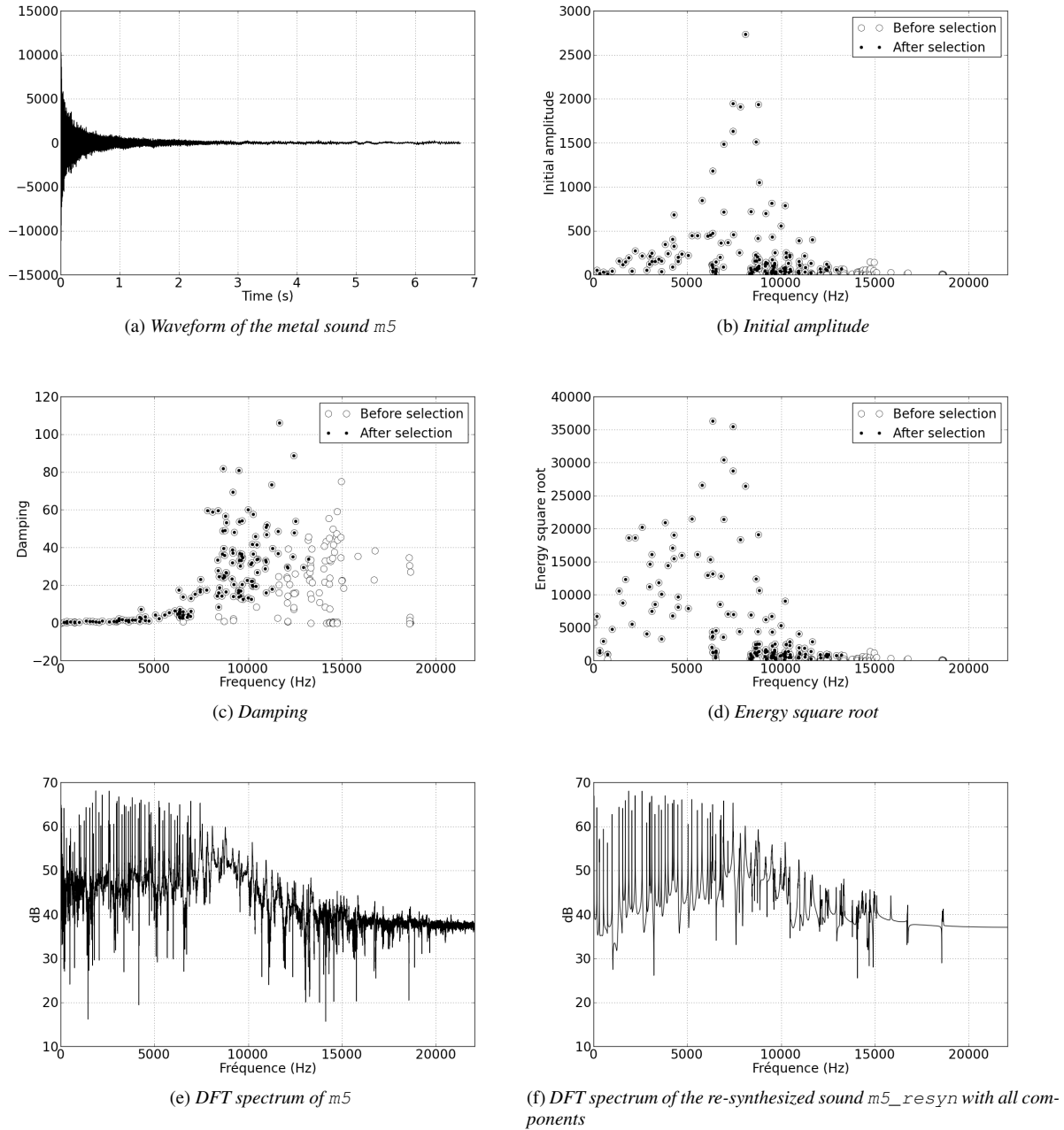


Figure 4: Overview of the analysis of m5 (a) using the ESPRIT algorithm over its Gabor transform. (b), (c) and (d) show the 246 mode parameters which have been initially extracted. (e) and (f) respectively show the DFT spectrum of the original sound m5 and the DFT spectrum of the re-synthesised sound m5_resyn; both sounds are available at [12]. The 152 modes marked with a black dot are the ones that remain after discarding the modes which initial amplitude is below the absolute detection threshold; the resulting synthesis sound m5_resyn_amp_ts can be listened to at [12].

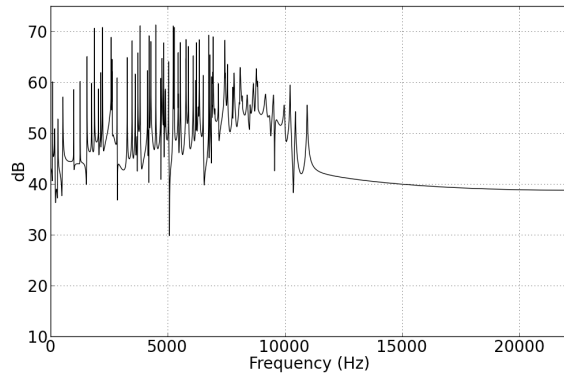


Figure 3: DFT spectrum of the re-synthesized sound *m5_std_esprit* obtained by applying a full band ESPRIT algorithm. The model order is $K = 206$.

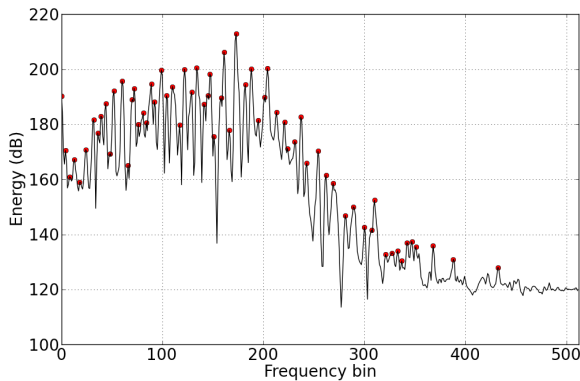


Figure 5: Energy of the Gabor transform of *m5* computed for each of its channels. The dots correspond to the channels identified as peaks.

reliable than in the full-band case, therefore the analysis order is computed for each of the selected channels, and set to the maximum of the ESTER criteria cost function. Doing so, a total number of 250 modes is obtained.

4.3. Discarding multiple components

If the distance between a set of channels on which an analysis has been performed is smaller than the bandwidth of the analysis window $g[l]$, the same component is likely to appear in all of these channels. These multiple estimations of the same component have to be identified, and only one will be kept for the final re-synthesis: the one which is the closest to the central frequency of its detection channel. In the example presented here, 4 components have been identified as replicas using a frequency confidence interval of 1 Hz. Fig. 4b, 4c and 4d show the mode parameters (amplitude, damping, energy as function of frequency) that remain after discarding the replicas. The resulting re-synthesized sound *m5_resyn* can be listen to on [12]. Fig. 4f shows the DFT spectrum of *m5_resyn* which can be compared to the DFT spectrum of the original analysed sound Fig. 4e.

4.4. Discarding irrelevant components

The estimated set of modes is the one that best fits the signal model (2) with respect to the Total Least Square criterion. However, as shown in Fig 4b, some of those modes are not relevant for they have an insignificant energy. In order to produce perceptually convincing sounds, one can rely on psychoacoustic results in order to discard inaudible modes. For instance, the absolute detection threshold can be used to discard modes by observing their initial amplitude. The black dotted modes on Fig. 4b, 4c and 4d represent the modes that remain after applying an absolute detection threshold ([13]) and setting the minimum of the threshold to the minimum amplitude that the sound format can handle (e.g. ± 1 for wav format coded as 16 bits integers). The resulting sound *m5_resyn_amp_ts*, containing 152 modes, can be listened to at [12].

It is also possible to use energy arguments and favour high energy modes over low energy modes. In the directory named ‘Cumulative synthesis’ available at [12] are stored successive re-synthesis of *m5* computed by successively adding the modes sorted in decrescent order of energy. One can note that there is no significant perceptual difference between the sounds beyond 105 modes.

5. FURTHER IMPROVEMENTS

One of the advantages provided by the use of time-frequency representations is the existence of efficient statistical estimators for the background noise. As it can be seen on Fig. 1, a significant number of Gabor coefficients describing an impact sound correspond to noise, and can therefore be used to estimate the variance of the stochastic part of the signal (see [11]). If the additive noise is coloured, it is even possible to estimate the variance in several selected frequency bands. Knowing the variance of the noise for each frequency channel offers the possibility to use noise masking properties of the human hearing to discard inaudible components, and possibly lead to a more selective criteria than the absolute detection threshold described in section 4.4.

The concept of nonstationary Gabor frames ([14]) makes it also possible to adapt the resolution of the Gabor transform so as to get an optimal compromise between precision and computational cost. It would allow, for instance, to take into account the logarithmical frequency resolution of the human hearing when applying the Gabor transform. Furthermore, it can be observed that the damping usually decreases with frequency; nonstationary Gabor frames would allow to adapt the time-step parameter of the Gabor frame along the frequency scale, so that computational cost is saved while a sufficient number of coefficients are taken for the analysis.

6. CONCLUSION

It has been shown that using the ESPRIT algorithm over time-frequency representations leads to perceptually convincing re-synthesis. The method has the same benefits than the sub-band analysis: it allows an extension of the analysis horizon, and it diminishes the complexity of the problem by only considering successive regions in the frequency domain; but on top of that, the information given by the time-frequency representation is of great use for targeting the analysis on the time-frequency intervals that contain the desired information. This avoids unnecessary analysis and reduces

the global computational cost.

7. REFERENCES

- [1] C. Verron, M. Aramaki, R. Kronland-Martinet, and G. Pallone, "A 3-d immersive synthesizer for environmental sounds," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 6, pp. 1550–1561, 2010.
- [2] "Sound Design Toolkit," <http://www.soundobject.org/SDT/>.
- [3] R. Schmidt, "Multiple emitter location and signal parameter estimation," *Antennas and Propagation, IEEE Transactions on*, vol. 34, no. 3, pp. 276–280, 1986.
- [4] R. Roy and T. Kailath, "ESPRIT-estimation of signal parameters via rotational invariance techniques," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 7, pp. 984–995, 1989.
- [5] M. Goodwin, "Matching Pursuits with Damped Sinusoids," in *Acoustics, Speech, and Signal Processing, 1997. Proceedings (ICASSP'97). IEEE International Conference on*. IEEE, 2004, pp. 2037–2040.
- [6] R. Badeau, *Méthodes à haute résolution pour l'estimation et le suivi de sinusoides modulées*, Ph.D. thesis, Ecole Nationale Supérieure des Télécommunications, 2005.
- [7] K. Ege, X. Boutillon, and B. David, "High-resolution modal analysis," *Journal of Sound and Vibration*, vol. 325, no. 4-5, pp. 852–869, 2009.
- [8] R. Badeau, B. David, and G. Richard, "A new perturbation analysis for signal enumeration in rotational invariance techniques," *Signal Processing, IEEE Transactions on*, vol. 54, no. 2, pp. 450–458, 2006.
- [9] S. Van Huffel, H. Park, and J.B. Rosen, "Formulation and solution of structured total least norm problems for parameter estimation," *Signal Processing, IEEE Transactions on*, vol. 44, no. 10, pp. 2464–2474, 1996.
- [10] K. Gröchenig, *Foundations of time-frequency analysis*, Birkhauser, 2001.
- [11] F. Millioz and N. Martin, "Estimation of a white Gaussian noise in the Short Time Fourier Transform based on the spectral kurtosis of the minimal statistics: Application to underwater noise," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5638–5641.
- [12] "Sample sounds," available at: <http://www.lma.cnrs-mrs.fr/~kronland/Dafx11/>.
- [13] T. Painter and A. Spanias, "Perceptual coding of digital audio," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 451–515, 2000.
- [14] Florent Jaillet, Peter Balazs, and Monika Dörfler, "Non-stationary Gabor frames," in *Proceedings of the 8th international conference on Sampling Theory and Applications (SAMPTA'09)*, Marseille, France, May 2009.

A PARAMETRIC MODEL OF PIANO TUNING

François Rigaud and Bertrand David,

Institut Telecom; Telecom ParisTech; CNRS LTCI;
Paris, France

firstname.lastname@telecom-paristech.fr

Laurent Daudet,

Institut Langevin; ESPCI Paristech;
CNRS UMR 7587; Paris, France
and Institut Universitaire de France

firstname.lastname@espci.fr

ABSTRACT

A parametric model of aural tuning of acoustic pianos is presented in this paper. From a few parameters, a whole tessitura model is obtained, that can be applied to any kind of pianos. Because the tuning of piano is strongly linked to the inharmonicity of its strings, a 2-parameter model for the inharmonicity coefficient along the keyboard is introduced. Constrained by piano string design considerations, its estimation requires only a few notes in the bass range. Then, from tuning rules, we propose a 4-parameter model for the fundamental frequency evolution on the whole tessitura, taking into account the model of the inharmonicity coefficient. The global model is applied to 5 different pianos (4 grand pianos and 1 upright piano) to control the quality of the tuning. Besides the generation of tuning reference curves for non-professional tuners, potential applications could include the parametrization of synthesizers, or its use in transcription / source separation algorithm as a physical constraint to increase robustness.

1. INTRODUCTION

One of the main factor that makes piano tuning so distinctive is the inharmonic nature of piano tones [1]. For a perfectly soft string, the spectrum of a note sound should be composed of purely harmonic partials. In practice, because of the stiffness of the piano wire, each partial is slightly sharper, and the higher the rank of the partial, the sharper the partial. This phenomenon directly affects the tuning because it constraints the tuner to stretch the intervals in order to cancel or control beats. Moreover, psycho-acoustical effects seem to be involved in the choice of the amount of stretching that is optimal according to the position in the tessitura [1] [2]. Due to the variations in piano scale designs and tuners' specific techniques, no single standard tuning rule can be established. However, some studies (see [3], [4]) have tried to formalize these rules used by tuners, to approximate aural tuning in a given range of the piano, taking into account inharmonicity measurements.

The purpose of this paper is to simulate aural tuning on the whole tessitura of a particular piano, based on the recordings of only a few isolated notes. This problem can be seen as an interpolation of inharmonicity and fundamental frequency across the whole tessitura, based on a limited set of initial data. In order to get a robust method, we constrain the interpolation with prior information on piano string design and tuning rules. This model can be used to generate tuning reference curves for non-professional tuners, to parametrize piano synthesizers, or be included as a constraint in transcription / source separation algorithms.

In Section 2, physical assumptions used to model the piano string vibration are given, and the relations between piano string

design and tuning are discussed. From this considerations, we propose in Section 3 a simple model with 2 parameters to represent the evolution of the inharmonicity coefficient on the whole tessitura. Then, in Section 4, we introduce a 4-parameter model based on tuning rules to generate reference tuning curves, by taking into account the inharmonicity model. We conclude (Section 5) with a discussion on potential applications of such model. For the sake of completeness, we describe in Appendix A the inharmonicity / fundamental frequency estimation algorithm that we used on single note recordings to obtain the reference values.

2. PHYSICAL CONSIDERATIONS IN PIANO TUNING

2.1. Physical modelling of piano string

Solving the transverse wave equation for a plain stiff string with fixed endpoints yields the following modal frequencies [1]:

$$f_n = nF_0\sqrt{1 + Bn^2}, \quad n \in \mathbb{N}^+ \quad (1)$$

where n is the mode index or partial rank, B the inharmonicity coefficient, and F_0 the fundamental frequency of a flexible string (with no stiffness). F_0 is related to the speaking length of the string L , the tension T and the linear mass μ according to:

$$F_0 = \frac{1}{2L} \sqrt{\frac{T}{\mu}} \quad (2)$$

Note that $F_0 \approx f_1$, but that strictly speaking this fundamental frequency is not directly measured as one peak in the spectrum: it is a global value of the tone that must theoretically be obtained from the whole set of partials. The stiffness is taken into account in B , with:

$$B = \frac{\pi^3 E d^4}{64 T L^2} \quad (3)$$

where E is the Young's modulus and d the diameter of the plain string. Note that this model is given for a string with fixed endpoints. It does not take into account the bridge coupling (with finite admittance), which modifies the partial frequencies, mainly in the low frequency domain [5], [6], [1], [7].

2.2. String set design influence on B

Piano strings are designed with the constraint to minimize the discontinuities in physical parameters variations [8], [9]. Three main discontinuities appear along the keyboard: the bass break between the bass and treble bridges, the transition between plain and wrapped strings and the transitions between adjacent keys having different

number of strings. The variations of B along the keyboard are mainly affected by the bass break which results in two main trends:

On the treble bridge, from C8 note downwards, B is decreasing because of the increase of L . Down to middle C (C4 note), the values of B are roughly the same for all the pianos and B follows a straight line in logarithmic scale [10]. This result is mainly due to the fact that string design in this range is standardized since it is not constrained by the limitation of the piano size.

To keep a reasonable size of the instrument, the bass bridge design reduces the growth of L . Then the linear mass of the string is increased in order to adjust the value of F_0 according to equation (2). Instead of increasing only the diameter d , which increases B and decreases the breaking strength, the strings are wrapped. Thus, on bass bridge, B is increasing from sharpest notes downwards. Note that the number of keys associated to the bass bridge and the design of their strings are specific for each piano.

2.3. Tuning influence on (F_0, B)

Most of the parameters in equations (2) and (3) are fixed at the string design. The only parameter the tuner can vary in order to adjust F_0 is the tension of the string T . In the same time, T affects the value of the inharmonicity constant B . Consequently, F_0 and B are dependent on each other because of physical relations and tuning considerations. In this paper, we assume that the relative variation of T during the tuning (of an initially slightly detuned piano) is small enough to consider that B remains constant.¹ It allows us to first extract a parametric model for B along the keyboard, and then to deduce tuning reference curves.

3. WHOLE TESSITURA MODEL FOR B

3.1. Parametric model

According to subsection 2.2, B should be modelled by two distinct functions corresponding to the two bridges, and could present a discontinuity at the bass break. In this paper we propose a “continuous” additive model on the whole tessitura, discretized for $m \in [21, 108]$, the midi note index from A0 to C8. We denote it by $B_\theta(m)$, with θ the set of parameters.

Usually, the evolution of B along the keyboard is depicted in logarithmic scale and presents two linear asymptotes. We denote by $b_T(m)$ (resp. $b_B(m)$) the Treble bridge (resp. the Bass bridge) asymptote of $\log B_\theta(m)$. Each asymptote is parametrized by its slope and its Y-intercept.

$$\begin{cases} b_T(m) = s_T \cdot m + y_T \\ b_B(m) = s_B \cdot m + y_B \end{cases} \quad (4)$$

According to [10], $b_T(m)$ is similar for all the pianos so s_T and y_T are fixed parameters. Then, the set of free (piano dependent) parameters reduces to $\theta = \{s_B, y_B\}$. $B_\theta(m)$ is set as the sum of the contributions of these two curves (4) in the linear scale:

$$B_\theta(m) = e^{b_B(m)} + e^{b_T(m)} \quad (5)$$

It should be emphasized that this additivity does not arise from physical considerations, but it is the simplest model that smoothes

¹For instance, if a note is increased by a quarter tone (50 cents) during the tuning, $\frac{\Delta F_0}{F_0} \simeq 2.9\%$. According to equations (2) and (3), $\frac{\Delta B}{B} = -2 \cdot \frac{\Delta F_0}{F_0} \simeq 5.9\%$.

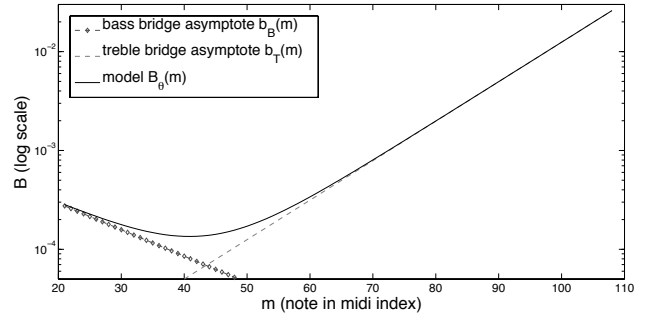


Figure 1: Model of B along the keyboard.

discontinuities between the bridges. Experimental data will show that it actually describes well the variations of B in the transition region around the two bridges. An example of this model for $B_\theta(m)$ is shown on Figure 1.

3.2. Parameter estimation

The results in this paper are obtained from single note recordings of 3 separate databases: Iowa² (1 grand piano), RWC [11] (3 grand pianos) and MAPS³ (1 upright piano). For a given note index $m \in [21, 108]$, $F_0^*(m)$ and $B^*(m)$ are the estimated values of $F_0(m)$ and $B(m)$ using the algorithm described in Appendix A.

We first estimate the fixed parameters $\{s_T, y_T\}$ using the data of all the pianos in the range C4–C8 ($m \in [60, 108]$, the standardized design range). These are obtained by a L1 linear regression (to reduce the influence of potential outliers) on the average of the estimated inharmonicity curves in logarithmic scale over the different pianos. We find $s_T \simeq 9.26 \cdot 10^{-2}$, $y_T \simeq -13.64$. These results are in accordance with estimates based on physical considerations [10]: $s_{T[10]} \simeq 9.44 \cdot 10^{-2}$, $y_{T[10]} \simeq -13.68$.

Finally, each piano is studied independently to estimate the particular parameters $\theta = \{s_B, y_B\}$ on a set of few notes M . θ is estimated minimizing the L1 distance between $B^*(m)$ and $B_\theta(m)$:

$$\theta^* = \arg \min_{\theta} \sum_{m \in M} |B^*(m) - B_\theta(m)| \quad (6)$$

We present on Figure 2 the curves of $B_\theta(m)$ obtained for every piano from a set of 3 quasi-equally spaced notes taken in the bass range A0–D3 ($m \in [21, 50]$). The discontinuity of the bass break is clearly observable for some pianos on the reference data curves (for instance between C#2 ($m = 37$) and D2 ($m = 38$) notes for the 2nd grand piano of the RWC database) and does not always occur at the same keys. The global variations are well respected. Note that some outliers are present (in the high treble range) in the whole tessitura data curves. This problem is discussed in the appendix and is due to the fact that for sharpest notes the partials are not in sufficient number to have a robust estimation of B . To evaluate the distance between the model and the whole tessitura data those outliers have been manually removed before the computation of the relative deviation between $B^*(m)$ and $B_\theta(m)$. We present on Figure 3 the histograms of the relative deviation computed in

²<http://theremin.music.uiowa.edu>

³<http://www.tsi.telecom-paristech.fr/aao/en/category/database/>

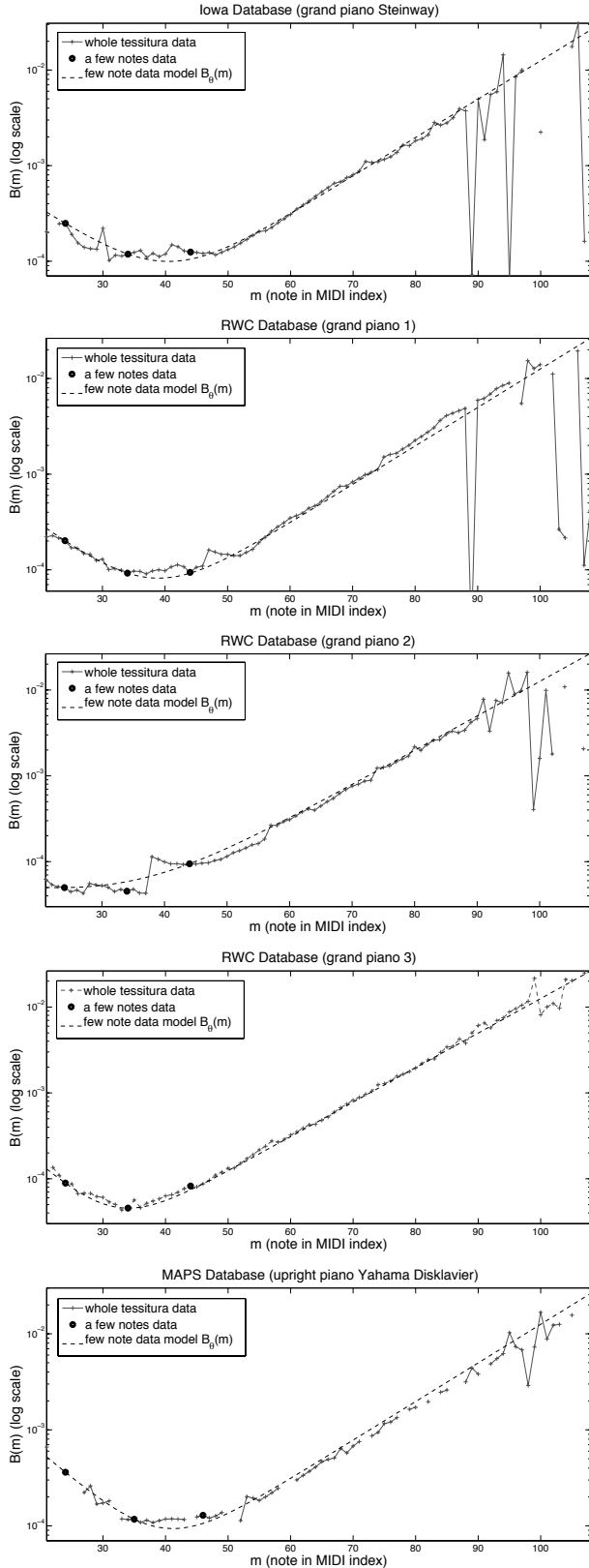


Figure 2: Superposition of the model $B_\theta(m)$ (estimated from 3 notes in the bass range) with the whole tessitura reference values for 5 different pianos.

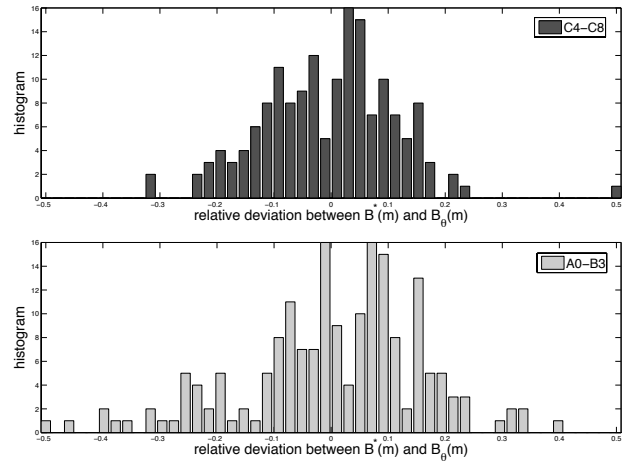


Figure 3: Histogram of the relative deviation between $B^*(m)$ and $B_\theta(m)$ computed for all the pianos in the range C4-C8 (at the top) and A0-B3 (at the bottom).

A0-B4 ($m \in [21, 71]$) and C4-C8 ($m \in [72, 108]$) ranges. In C4-C8 range, the mean and the standard deviation are respectively equal to $-4.2 \cdot 10^{-3}$ and $1.16 \cdot 10^{-1}$. In A0-B4 range we have respectively $4.6 \cdot 10^{-3}$ and $1.57 \cdot 10^{-1}$.

4. WHOLE TESSITURA MODEL FOR F_0

4.1. Aural piano tuning principles

Every tuning begins by the tuning of the reference note, in most cases the A4 at 440Hz. To do so, the tuner adjusts the tension of the strings to cancel the beats produced by the difference of frequency of the tuning fork (a quasi perfect sinusoid) and the first partial of the note. Thus, $f_1(m = 69) = 440\text{Hz}$.

From A4, the tuner builds the reference octave F3-F4 according to the equal temperament in controlling (or counting) the beats of different intervals (for instance between the 3rd partial of a reference note and the 2nd of its fifth) [12]. For high inharmonicity pianos the frequency deviation between the first partial of the note and the theoretical fundamental frequency given by the equal temperament in this octave can be about ± 8.6 cents ($\pm 0.5\%$) [3].

Finally, from this reference octave in the middle of the tessitura, each note is tuned step by step with the same procedure. Because of the partial deviation due to the inharmonicity, the octaves are stretched to more than a 2:1 frequency ratio. For a reference note of midi index m , $f_1(m + 12) > 2f_1(m)$ because $f_2(m) > 2f_1(m)$. Moreover, the amount of stretching of the octaves in the different parts of the keyboard is linked to psychoacoustic effects and tuner's personal tastes. It is well known that the piano sounds better in the bass range if the amount of stretching is more important than in the treble range (even if the inharmonicity effect is less important). This fact is linked to the underlying choice of the type octave during the tuning [2]. For instance, in a 4:2 type octave, the 4th partial of the reference note is matched to the 2nd partial of its octave. Depending on the position in the tessitura, the piano can be tuned according to different type octaves: 2:1, 4:2, 6:3, 8:4, ... or a compromise of two. This means that the tuner may not be focused only on cancelling beats between a pair

of partials, but that he controls an average beat generated by a few partials of the two notes.

Here, we propose a 4-parameter model for synthesizing aural tuning on a given piano. The steps may be a simplified version of those done by a tuner but the global considerations (stretching inherent in the inharmonicity and the type octave choice) are taken into account. We begin by the A4 reference note, setting $f_1(m = 69) = 440\text{Hz}$. Then, we introduce (Subsection 4.2) a 3-parameter model to estimate the tuning of all the A keys from the A4. In Subsection 4.3, we propose a model to tune all the notes inside of a fixed octave interval (for instance A4-A5 previously determined). Finally, in Section 4.4, we introduce 1 extra parameter to take into account a global detuning and we present the results for the 5 pianos. Note that the following expressions are established for upper interval construction, but the same reasoning can be applied for lower intervals.

4.2. Octave interval tuning

4.2.1. Model

During the tuning of an upper octave interval, the cancellation of the beats produced by the u -th partial of a reference note indexed $m - 12$ and the v -th partial of its octave indexed m ($u = 2v$) can be done by tuning $F_0(m)$ such as⁴:

$$F_0(m) = F_0(m - 12) \cdot \frac{u\sqrt{1 + B(m - 12)u^2}}{v\sqrt{1 + B(m)v^2}} \quad (7)$$

The choice of the type octave is parametrized by introducing the variable $\rho \in \mathbb{N}^+$, such as $u = 2\rho$ and $v = \rho$. Usually the maximal value for ρ is 6 (it corresponds to a 12:6 type octave which can sometimes occur in the low bass range of grand pianos). We denote by $\rho_\varphi(m)$ the model of ρ on the whole tessitura given for a set of parameter φ . Then,

$$F_0(m) = 2 F_0(m - 12) \sqrt{\frac{1 + B(m - 12) \cdot 4\rho_\varphi(m)^2}{1 + B(m) \cdot \rho_\varphi(m)^2}} \quad (8)$$

This model takes into account the cancellation of the beats produced by a single pair of partials. In practice, the deviation $\frac{F_0(m)}{2F_0(m-12)}$ should be a weighted sum of the contribution of two pairs of partials, because the amount of stretching may result from a compromise between two type octaves. An alternative model to take into account this weighting is to allow non-integer values for $\rho_\varphi(m) \in [1, +\infty[$. For example, if the octave tuning of a note indexed m is a compromise between a 2:1 and 4:2 type octaves, $\rho_\varphi(m)$ will be in the interval $[1, 2]$. This model loses the physical meaning ($u = 2\rho$ and $v = \rho$ are not anymore related to partial ranks), but presents the advantage to be easily inverted to estimate $\rho_\varphi(m)$. Note that this model for octave interval tuning could be generalized to other intervals tuning by considering the beats inherent in the equal temperament. Indeed, in equal temperament only octave intervals can have consonant partials.

⁴Note that F_0 is defined as being the fundamental frequency for a perfectly soft string. In practice it is not present in the piano tone so the tuner adjusts f_1 , the frequency of the first partial. F_0 is used in the equations of this section because it is more practical to manipulate. In the end, equation (1) is applied to obtain $f_1(m) = F_0(m)\sqrt{1 + B(m)}$.

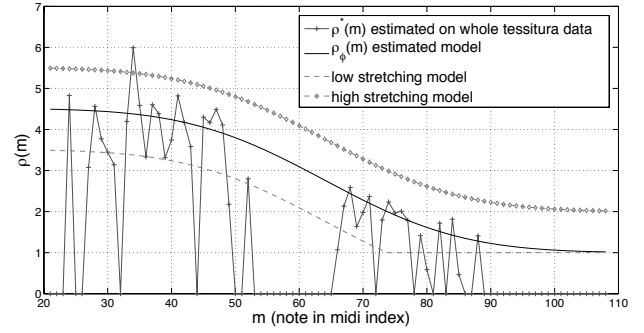


Figure 4: Superposition of $\rho^*(m)$ averaged from 3 pianos, and the model $\rho_\varphi(m)$. High and low stretching curves are respectively, arbitrarily defined by $\rho_\varphi(m) + 1$ and $\max(\rho_\varphi(m) - 1, 1)$.

4.2.2. Estimation of $\rho_\varphi(m)$

We choose to model $\rho_\varphi(m)$ as follows:

$$\rho_\varphi(m) = \frac{K}{2} \cdot \left(1 - \operatorname{erf}\left(\frac{m - m_0}{\alpha}\right) \right) + 1 \quad (9)$$

with erf the error function. It expresses the fact that the amount of stretching inherent in the type octave choice is decreasing from the low bass range to the high treble range and that it is limited by horizontal asymptotes at each extremity. The set of parameters is then $\varphi = \{m_0, \alpha, K\}$. m_0 is a parameter of translation along m . α rules the slope of the decrease. K settles the value of the low bass asymptote. Note that in (9) the high treble asymptote is set to 1 because it corresponds to the minimal type octave (2:1).

$\rho^*(m)$ is estimated on the data $F_0^*(m)$ and $B^*(m)$ by inverting equation (8):

$$\rho^*(m) = \sqrt{\frac{4F_0^*(m - 12)^2 - F_0^*(m)^2}{F_0^*(m)^2 B^*(m) - 16F_0^*(m - 12)^2 B^*(m - 12)}} \quad (10)$$

Then, the set of parameters is estimated minimizing the L1 distance between $\rho_\varphi(m)$ and $\rho^*(m)$ on a set M of notes.

$$\varphi = \arg \min_{\varphi} \sum_{m \in M} |\rho^*(m) - \rho_\varphi(m)| \quad (11)$$

Finally, $\rho^*(m)$ has been estimated for the 3 best tuned pianos of the database (the selection criterion was that their tuning deviation from equal temperament is following the global variations of the Raylsback theoretical curve [1]), and averaged to obtain a mean curve from different tuners.⁵ The parameter estimation gives $m_0 \simeq 64$ (the curve is centred on the middle octave), $\alpha \simeq 24$, and $K \simeq 4.51$ (in the low bass range, the tuning is a compromise between 8:4 and 10:5 type octaves). The results are depicted on Figure 4. Some values of $\rho^*(m)$ are missing in the treble and the bass range because we removed the outliers from the estimation of $B^*(m)$ and $F_0^*(m)$. Because F3-F4 ($m \in [53, 65]$) is the reference octave of the tuning, $\rho^*(m)$ is not estimated on it. From

⁵In practice the estimation of $\rho^*(m)$ could be done for each piano to model their actual tuning. We choose here to obtain a reference stretching curve from well-tuned pianos in order to control the tuning of the 5 pianos in Subsection 4.4. In this case the application is not anymore the interpolation of the tuning on the whole tessitura of each piano.

$\rho_\varphi(m)$ defined as a mean stretching model, we define arbitrarily a high stretching model by $\rho_\varphi(m) + 1$ and a low stretching model by $\max(\rho_\varphi(m) - 1, 1)$. The low stretching model is saturated to 1 in the treble range because $\rho \in [1, +\infty[$.

4.3. Model for semitone tuning in a given octave interval

Once the octave intervals are built according to equation (8), the whole tessitura is interpolated semitone by semitone. If there was no stretching, the semitones would be equally spaced by the ratio of $\sqrt[12]{2}$ given by the equal temperament. In practice, the frequency ratio between 2 adjacent notes is a little higher than $\sqrt[12]{2}$. We model this deviation as follows:

$$f_1(m+1) = f_1(m) \sqrt[12]{2 + \varepsilon(m+1)} \quad (12)$$

with $\varepsilon \ll 1$. As a first order model, we assume that ε varies linearly with B . This dependence underlines the fact that the higher B , the higher the deviation should be. Thus,

$$\varepsilon(m+1) = \lambda \cdot B(m+1) \quad (13)$$

λ is estimated in the given octave interval and takes into account the stretching related to the type octave through the previous estimation of $f_1(m+12)$. Recursively we have:

$$f_1(m+12) = f_1(m) \prod_{p=1}^{12} \sqrt[12]{2 + \lambda \cdot B(m+p)}$$

By taking the logarithm, and developing at the first order, λ can be estimated by:

$$\lambda = \frac{24 \log(f_1(m+12)/2f_1(m))}{\sum_{p=1}^{12} B(m+p)} \quad (14)$$

4.4. Global detuning and results

Once the tuning has been estimated on the whole tessitura, the real piano tuning can present a slight global detuning compared to the model $f_1(m)$. The detuning or deviation of each note from the equal temperament (ET) is given in cents by:

$$d_1(m) = 1200 \log_2 \frac{f_1(m)}{F_{0ET}(m)} \quad (15)$$

with

$$F_{0ET}(m) = 440 \cdot 2^{(m-69)/12} \quad (16)$$

We introduce the global detuning through the 4th parameter d_g , which is estimated by minimizing the L1 distance, on the reference octave F3-F4 ($m \in [53, 65]$) between $d_1^*(m)$, the detuning estimated on data, and $d_1(m) + d_g$ the detuning of the model:

$$d_g = \arg \min_{d_g} \sum_{m=53}^{65} |d_1^*(m) - (d_1(m) + d_g)| \quad (17)$$

Finally, Figure 5 shows the deviation from ET of the estimated models for the 3 amounts of stretching (mean, low and high) applied to the 5 pianos. Comparing the curves of the model and the data, we can see that RWC2 and RWC3 piano seem well-tuned. On the contrary, the tuning of RWC1 piano is not stretched in the bass range and the tuning of Iowa and MAPS pianos should be a little more stretched in the treble range, according to our model. Further research and discussions with piano tuners will investigate whether this discrepancy is indeed due to an inappropriate tuning, or a limitation of our model.

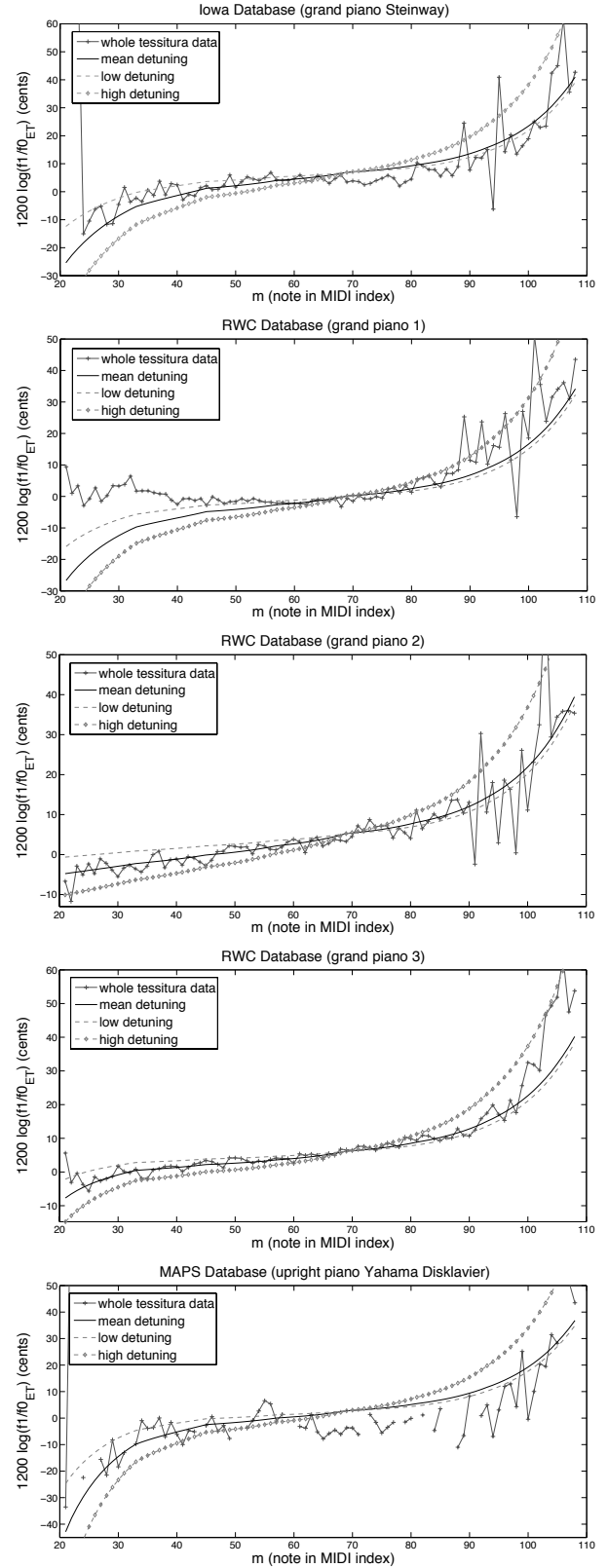


Figure 5: Superposition of the deviation from ET of the model $f_1(m)$, with the whole tessitura data for 5 pianos. $f_1(m)$ is computed for the 3 models of $\rho_\varphi(m)$ depicted on the figure 4 and corresponding to a mean, low and high octave stretching.

5. CONCLUSIONS

We presented in this paper a model composed of a few parameters for aural tuning on the whole piano tessitura. The parameters can be learned from the estimation of the inharmonicity coefficient and the fundamental frequency of a few single note recordings. For the inharmonicity coefficient, 3 notes in the bass range are sufficient to obtain a good interpolation on the whole tessitura. For the fundamental frequency, a few more notes are needed on the whole tessitura. The model takes into account physical considerations of the piano string scale design and piano tuning rules used by tuners.

It is intended to be useful for controlling the tuning of a given piano (as shown in Subsection 4.4) or for parametrizing the tuning of physically-based piano synthesizers. Following the steps proposed in this paper it could be possible to generate an inharmonicity curve specific to a given piano (or to set the inharmonicity coefficient of each note if the design of the target piano is perfectly known), choose the amount of stretching on the whole tessitura and an eventual global detuning to automatically generate an appropriate tuning.

The next step of this work is to include this model as constraints in multipitch (such as [17]) or automatic transcription algorithms of piano music. Instead of searching for 88 independent values of the inharmonicity coefficient and of the fundamental frequency, it strongly constraints the estimation to only 6 parameters, which should result in increased robustness.

6. ACKNOWLEDGMENTS

This research was partially funded by the French Agence Nationale de la Recherche, PAFI project.

A. APPENDIX: (F_0, B) ESTIMATION ON SINGLE NOTES

The problem of estimating (F_0, B) on single piano note recordings has been dealt with by several authors, amongst these: [13], [14], [15], [4]. Each of these algorithms could potentially be used for the estimation of the reference data used by the tuning model presented in the body of the article. However, the algorithm below comprises a new preprocessing stage of adaptive noise level estimation, which avoids most of potential outliers during the partial selection. Satisfactory results are usually obtained up to the C6 note.

Algorithm: The main idea is to perform a linear regression on an alternative version of the inharmonicity law (1):

$$\frac{f_n^2}{n^2} = F_0^2 + F_0^2 B \cdot n^2 \quad (18)$$

This equation is linear according to n^2 . If we collect the f_n frequencies in the spectrum $S(f)$ and we know their rank n , we just have to do a linear regression to obtain F_0 and B . We use Least Absolute Deviation Regression (LADR) to discard outliers (phantom partials or partials affected by strong bridge coupling inharmonicity). The main steps of the algorithm are presented on Figure 6. The input is the magnitude spectrum $S(f)$ computed with zero padding on 2^{16} frequency bins from a 500ms window in the decay part of the sound. The first step is a noise level $NL(f)$ estimation of the magnitude spectrum. This preprocessing stage allows the separation of spectral peaks related to partials from noise. Then, the partials above the noise level corresponding to transverse

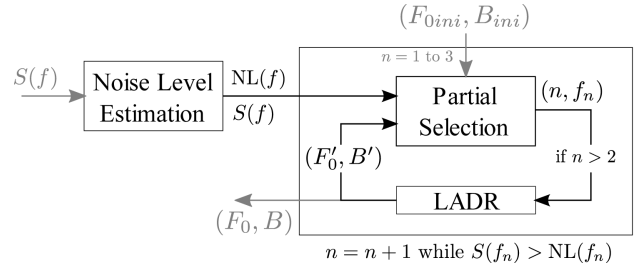


Figure 6: Estimation of (F_0, B) . Algorithm scheme.

modes of vibration are picked up by an iterative process, estimating intermediate (F_0, B) values at each step.

Noise level estimation: We assume that the noise is an additive colored noise, i.e. generated by a filtered white gaussian noise [16]. In a given narrow band, if the filters have a quasi flat frequency response the noise can be considered as white gaussian, and its spectral magnitude follows a Rayleigh distribution:

$$p_X(x; \sigma) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)} \quad (19)$$

In this pre-processing stage, we want to estimate the noise distribution in each band without removing the partials. To do so, a good estimator for σ is the median $med = \sigma \sqrt{\ln(4)}$. Indeed, in a given narrow band there are much less bins corresponding to partials than bins corresponding to noise, so partials have a reduced influence in the estimate of the noise median. The tradeoff sits in the choice of the bandwidth: the bands have to be narrow enough so that the white noise approximation holds, but wide enough so that most of the bins correspond to noise. We chose a 300Hz median filtering on the magnitude spectrum $S(f)$ to estimate $\sigma(f)$. Finally, we define the noise level in each band $NL(f)$ as the magnitude such that the cumulative distribution function is equal to a given threshold T , set to $T = 0.9999$. With this choice of T , only 6 bins corresponding to noise on average (out of 2^{16}) should be above the noise level. The cumulative density function of a Rayleigh distribution is given by:

$$c_X(x; \sigma) = 1 - e^{-x^2/(2\sigma^2)} \quad (20)$$

Partial selection: The f_n are extracted in the same time as their rank n by an iterative process. We begin with an approximative value of $F_0 = F_{0ini}$ given by equal temperament (the processed note is supposed to be known) and with $B_{ini} = 0$, for the search of the first three partials. Then, for each iteration we perform LADR according to equation (18) to estimate an intermediate (F_0', B') couple which will help in selecting the next partial. Each f_n frequency partial is searched in the range $nF_0'\sqrt{1+B'n^2} + [-\frac{F_{0ini}}{5}, \frac{F_{0ini}}{5}]$. The width of the search interval is set empirically. Once no partial is found above the noise level the algorithm terminates. The last iteration is presented on Figure 7.

Influence of the dynamics: In practice, for a given note, sound spectra can significantly vary according to dynamics. For *forte* dynamics, a lot of “phantom” partials can appear in the spectrum (non-linear coupling of transverse waves with longitudinal waves), be picked during the partial selection and corrupt the linear regression. Another limitation can appear in *piano* dynamics: for sharp notes (from C6 to C8) the transverse mode partials are too weak and not in sufficient number to correctly process the selection and the regression steps.

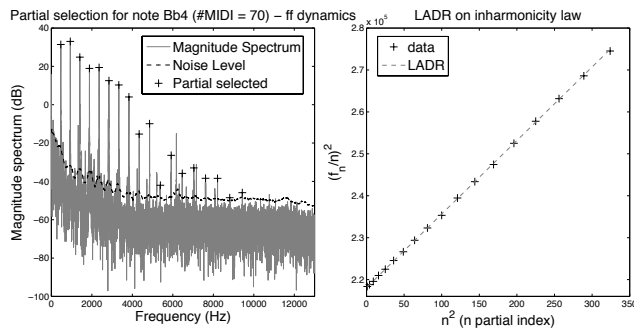


Figure 7: Partial selection and LADR at the last iteration of the algorithm.

B. REFERENCES

- [1] N.H. Fletcher and T.D. Rossing, *The physics of musical instruments*. 2nd Ed., pp. 352–398, Springer, 1998.
- [2] B. Bremmer, “Aural tuning tests for 2:1, 4:2 and 6:3 type octaves,” Tech. Rep., Piano Technicians Guild, 2007.
- [3] J. Lattard, “Influence of inharmonicity on the tuning of a piano,” *J. Acoust. Soc. Am.*, vol. 94, 1993.
- [4] L.I. Ortiz-Berenguer, F.J. Casajús-Quirós, M. Torres-Guijarro, and J.A. Beracoechea, “Piano transcription using pattern recognition: aspects on parameter extraction,” in *Proc. DAFx-04*, Oct. 2004.
- [5] G. Weinreich, “Coupled piano strings,” *J. Acoust. Soc. Am.*, vol. 62, no. 6, 1977.
- [6] C. E. Gough, “The theory of string resonances on musical instruments,” *Acta Acustica with Acustica*, vol. 49, no. 2, 1981.
- [7] A. Chaigne and J. Kergomard, *Acoustique des instruments de musique*, pp. 235–253, Belin, 2008, ISBN 978-2-7011-3970-8.
- [8] Jr. H.A. Conklin, “Design and tone in the mechanoacoustic piano. Part III. Piano strings and scale design,” *J. Acoust. Soc. Am.*, vol. 100, 1996.
- [9] A. Stulov, “Physical modelling of the piano string scale,” *EURASIP Journal on Applied Signal Processing*, pp. 977–984, 2008.
- [10] R.W. Young, “Inharmonicity of plain wire piano strings,” *J. Acoust. Soc. Am.*, vol. 24, 1952.
- [11] M. Goto, T. Nishimura, H. Hashiguchi, and R. Oka, “Rwc music database: Music genre database and musical instrument sound database,” in *ISMIR*, 2003, pp. 229–230.
- [12] B. Bremmer, “Midrange piano tuning,” Tech. Rep., Piano Technicians Guild, 2007, www.billbremmer.com.
- [13] J. Rauhala, H.M. Lehtonen, and V. Välimäki, “Fast automatic inharmonicity estimation algorithm,” *JASA Express Letters*, vol. 121, 2007.
- [14] A. Galembo and A. Askenfelt, “Signal representation and estimation of spectral parameters by inharmonic comb filters with application to the piano,” *IEEE Trans. on Speech and Audio Processing*, vol. 7, pp. 197–203, March 1999.
- [15] M. Hodgkinson, J. Wang, J. Timoney, and V. Lazzarini, “Handling inharmonic series with median-adjustive trajectories,” in *Proc. DAFx-09*, Sept. 2009.
- [16] C. Yeh and A. Röbel, “Adaptive noise level estimation,” in *Proc. DAFx-06*, Sept. 2006.
- [17] V. Emiya, R. Badeau, and B. David, “Multipitch estimation of piano sounds,” *Trans. Audio, Speech and Lang. Proc.*, vol. 18, 2010.

AUDIO DE-THUMPING USING HUANG'S EMPIRICAL MODE DECOMPOSITION

Paulo A. A. Esquef*

Coordination of Systems and Control
National Lab. of Scientific Computing - MCT
Petrópolis, Brazil
pesquef@lncc.br

Guilherme S. Welter

Coordination of Systems and Control
National Lab. of Scientific Computing - MCT
Petrópolis, Brazil
gswelter@lncc.br

ABSTRACT

In the context of audio restoration, sound transfer of broken disks usually produces audio signals corrupted with long pulses of low-frequency content, also called thumps. This paper presents a method for audio de-thumping based on Huang's Empirical Mode Decomposition (EMD), provided the pulse locations are known beforehand. Thus, the EMD is used as a means to obtain pulse estimates to be subtracted from the degraded signals. Despite its simplicity, the method is demonstrated to tackle well the challenging problem of superimposed pulses. Performance assessment against selected competing solutions reveals that the proposed solution tends to produce superior de-thumping results.

1. INTRODUCTION

Severe damages or discontinuities to the grooves of a disk, such as those produced by deep scratches or breakages, may give rise to long-duration pulses of low-frequency content in the resulting audio signal, during disk playback [1, 2]. Being typically preceded by high-amplitude impulsive disturbances, long pulses are considered the impulse response of the stylus-arm system added to the waveform of interest [1, 2].

An illustration of a synthetically generated pulse is seen in Figure 1. As can be seen, the pulse waveform seems to have both an amplitude-modulated component (decaying exponential envelope) and a frequency-modulation component. Typically, the pulse oscillations start at about 150 Hz, right after the initial click, and decay exponentially down to about 10 Hz.

To the author's knowledge, apart from crude high-pass filtering, which is usually unsatisfactory, there are three available techniques for long pulse removal or audio de-thumping. Brief descriptions of these methods follow.

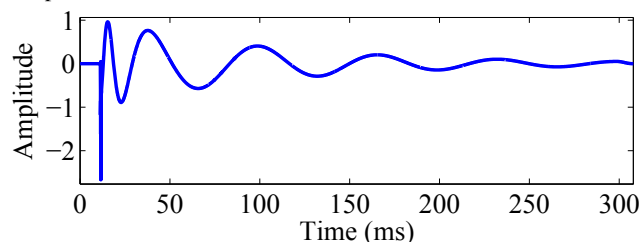


Figure 1: Synthetic example of a long pulse. The initial click occurs at about 11 ms.

The so-called template matching method has been first proposed by Vaseghi in [1] and [3]. Its main assumption is that long

pulses are shape-invariant, for being the impulse response of a given stylus-arm system. Thus, if a clean version of the pulse (a template) is available, its time-reversed version can be used as a matched filter to detect other pulse occurrences in the audio signal. Furthermore, amplitude-scaled versions of the template can be used to suppress corrupting pulses from the signal by simple subtraction. The remaining initial click is supposed to be removed afterward by standard de-clicking techniques [2].

In [4, 2, 5] Godsill and colleagues have proposed a model-based signal separation technique for audio de-thumping. The first step of the method consists in estimating two distinct autoregressive (AR) models: one of high order for the signal of interest and another of low-order for the corrupting pulse. Then, pulse removal is achieved by separation of the two AR processes. In this approach, the initial click is taken as part of the long pulse, being modeled by the same AR model of the pulse, but with a much higher excitation variance. Therefore, suppression of the initial click and the pulse is taken care of at once by the method.

Simple non-linear filtering techniques for audio de-thumping have been proposed in [6] by Esquef and colleagues. In this solution, hereafter referred to as the TPSW method, an initial estimate for the long pulse is obtained from a non-linear filtering technique called two-pass split window (TPSW) [7, 6], which is capable of producing relatively smooth pulse estimates, despite the presence of the high-amplitude clicks that precede the pulses. Then, these pulse estimates are made even smoother by means of an overlap-and-add signal segmentation with low-order polynomial fitting.

Thorough comparisons among the aforementioned methods is beyond the scope of this paper. Nevertheless, in general terms, the main advantages of the template matching method are its simplicity and ability to detect long pulses even if the initial clicks are absent. However, the solution is less flexible to tackle more challenging situations, such as the occurrence of superimposed long pulses, i.e., when a second pulse appears within the duration of a preceding one. According to the results presented in [6] the AR-separation and TPSW methods perform equally well, being the latter less intense computationally. Moreover, both can treat superimposed pulses.

In this paper an alternative solution to audio de-thumping is proposed. More specifically, it makes use of the so-called Empirical Mode Decomposition (EMD) [8] and its improved version, the Complementary Ensemble EMD (CEEMD) [9, 10], as a means to obtain estimates of long pulses corrupting an audio signal of interest. In principle, the EMD is capable of decomposing a given time-domain waveform into a finite set of Intrinsic Mode Functions (IMFs) and a monotonic residue, each IMF being a single AM-FM component. Since a long pulse can be well characterized as a single AM-FM component, the choice of the EMD for the

* The work of Dr. Esquef was supported by CNPq-Brazil via grants no. 472856/2010-3 and 306607/2009-3.

problem at hand seems justified.

The experimental results reported in this paper reveal that the EMD and the CEEMD are effective and simple tools to provide adequate pulse estimates. Performance evaluation of the proposed method against the AR-separation and TPSW methods is carried out via the Perceptual Audio Quality Measure (PAQM) [11]. The attained results show that the CEEMD-based audio de-thumping performs comparably to the competing solutions.

The remainder of the paper is organized as follows. In Section 2 brief reviews of the EMD and the CEEMD are given. The proposed pulse estimation method is explained in Section 3. The experimental setup defined for the comparative tests is described in Section 4. In Section 4.3 the attained results are presented and discussed. Finally, conclusions are drawn in Section 5.

2. THE EMPIRICAL MODE DECOMPOSITION

The Empirical Mode Decomposition was originally introduced by Huang and collaborators [8] as a way to decompose multicomponent signals into constituent functions from which meaningful instantaneous frequencies could be estimated via the Analytical Signal approach [12]. The EMD decomposition does not assume any basis function since it is an entirely data-driven iterative algorithm that operates over signal envelopes.

The EMD method decomposes a signal into components called *Intrinsic Mode Functions* (IMFs), which are typically characterized by zero-mean oscillations modulated by a slowly varying envelope. An IMF must have the following properties [8, 13, 14]:

1. The number of extrema and the number of zero-crossings must be either equal or differ at most by one;
2. The arithmetic mean between the upper and lower envelopes of an IMF must be zero at any point of its domain.

With reference to the item 2 above, the upper (lower) envelope is usually obtained via low-order polynomial fitting to the local maxima (minima) of the signal. Variations on the EMD algorithm exist [14, 15, 16] and are mainly concerned with two issues: different criteria to stop the intermediate iterative sifting procedure that culminate in an acceptable IMF; and alternative data extrapolation schemes to obtain the signal envelopes [17, 10].

2.1. EMD Implementation

For the experiments reported in the paper, a standard version of the EMD, *i.e.*, one that uses a Cauchy-type stopping criterion and natural cubic spline interpolation to compute the envelopes [8, 13], has been implemented in Matlab. Alternatively, these envelopes can also be obtained via piecewise cubic Hermite interpolating polynomials across local maxima (minima).

Considering a signal $x(t)$, the EMD is described as follows.

1. Let $j = 1$ and set $x_j(t) = x(t)$;
2. Identify all local maxima and minima of $x_j(t)$;
3. Obtain the upper envelope $e_{\text{upper}}(t)$ (respectively, lower envelope $e_{\text{lower}}(t)$) by polynomial interpolation across the local maxima (respectively, local minima) of $x_j(t)$;
4. Compute the mean envelope $m(t) = [e_{\text{upper}}(t) + e_{\text{lower}}(t)]/2$;
5. Obtain an IMF estimate $C_j(t) = x_j(t) - m(t)$;

6. If $m(t)$ is a non-monotonic function (or if it has enough extrema to allow envelope computation), make $x_j(t) = m(t)$; increment j by one unit; and go back to step 2 to obtain the subsequent IMFs. Otherwise, stop the iterations and set the residue of the decomposition as $r(t) = m(t)$.

In practice, step 5 above is insufficient to produce a proper IMF. To remedy this, step 5 is modified to include an inner loop to perform additional siftings to $x_j(t)$. More specifically, a so-called proto-IMF is defined as $C_{j,k}(t) = x_j(t) - m(t)$ for the k^{th} iteration of the sifting loop, which must continue until a stopping criterion (defined below) is satisfied. If an additional sifting is needed, then one sets $x_j(t) = C_{j,k}(t)$ and returns to step 2 going through step 5. The final IMF estimate $C_j(t)$ is then obtained as the last $C_{j,k}(t)$ of the sifting loop.

Here, the chosen stopping criterion is the same adopted in [13], *i.e.*, the iterations stop when the quantity

$$S_d = \text{Var}\{C_{j,k-1}(t) - C_{j,k}(t)\} / \text{Var}\{C_{j,k-1}(t)\} \quad (1)$$

becomes smaller than a pre-assigned value, typically within the range [0.0001–0.0003]. After obtaining a total of J IMFs and a residual trend $r_J(t)$, the original signal can be reconstructed by summing up all IMFs and the trend: $x(t) = \sum_{j=1}^J C_j(t) + r_J(t)$.

For broad spectrum signals such as those of fractal or Gaussian processes, the maximum number of IMFs is approximately $\log_2 L$, where L is the number of samples of the signal [18].

2.2. Complementary Ensemble EMD

EMD has been successfully used for analysis of diverse kinds of signals, mainly due its ability to tackle responses of non-linear and non-stationary systems. Nevertheless, the presence of intermittency in such signals often results in a phenomenon called *mode mixing* [13, 14], where coherent parts of the signal may end up in adjacent IMFs, thus devoid of physical meaning.

The original EMD algorithm is sensitive to the addition of small perturbations to the input signal, in the sense that it may produce a new set of IMFs in comparison with those of the noiseless version. Based on this fact, Wu and Huang [10] proposed that more reliable IMFs should be estimated from EMD of an ensemble formed by a given input signal artificially corrupted with several realizations of Gaussian noise.

The idea behind of the Ensemble EMD (EEMD) is to take a large number of noisy versions of the original signal

$$x^{(i)}(t) = x(t) + \varepsilon w^{(i)}(t), \quad (2)$$

where $\varepsilon w^{(i)}(t)$ is the i -th realization of a zero-mean white Gaussian noise with standard deviation ε , which can be made a fraction of that of $x(t)$. After obtaining the IMFs $C_j^{(i)}(t)$ for each realization $x^{(i)}(t)$, the final result is obtained by averaging the IMFs across all realizations:

$$\tilde{C}_j(t) = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N C_j^{(i)}(t). \quad (3)$$

Given a non-infinitesimal ε and a large enough N , the residual noise amplitude will be proportional to ε/\sqrt{N} and the resulting IMFs stable. More importantly, the EEMD largely reduces the mode mixing problem [10], thus improving the EMD performance at the expense of a much higher computational load.

A further improvement to the EEMD is the Complementary EEMD (CEEMD), in which the ensemble is formed by $N/2$ complementary pairs of noise realizations with symmetric amplitude. This way, Eq. (2) is modified to $x^{(i)}(t) = x(t) + (-1)^i \varepsilon w^{(i-\gamma)}(t)$, for $i = 1, 2, \dots, N$, where $\gamma = i \bmod 2$. The IMFs of the thus constructed ensemble are then obtained as before via Eq. (3). This procedure ensures that the residual noise will be zero.

3. LONG PULSE ESTIMATION VIA EMD AND CEEMD

Similar to the AR-separation and TPSW methods, the proposed EMD-based de-thumping requires prior knowledge of pulse locations in time. This means that, for a particular pulse, estimates of its onset time and duration must be available. In practice, the former can be inferred from the location of the initial click, which can easily be obtained by standard detection techniques [2].

From the auditory perception perspective, the most salient part of a long pulse is its beginning, for its higher amplitude and frequency. Therefore, pulse duration estimates can be obtained by visual inspection. In other words, underestimation of pulse durations is likely to produce no audible effects.

3.1. EMD-based Estimation of Single Pulses

For didactic reasons, EMD-based estimation of single pulses is presented first, being that of superimposed pulses left to later.

The main steps of the pulse estimation procedure (one pulse at a time) are listed below. More specific details of each step are given in the sequel.

1. Select a portion of the signal of interest containing one single long pulse (to be called input signal hereafter);
2. Extend the input signal backward in time;
3. Analyze the extended input signal via the EMD. The main parameter to be defined is the maximum number of IMFs.
4. Form the pulse estimate by mixing together partial reconstructions of the signal, with different levels of detail, via an overlap-and-add windowing scheme.

As regards step 1, the beginning of the input signal should coincide with that of the pulse, i.e., it should start right after the initial click. The duration of the segment should be approximately that of the observed long pulse. It is advisable though to add about 5 ms to the duration in order to overcome boundary effects that may affect IMF estimation [17, 10]. For that very reason, step 2 is taken. The idea here is to analyze an input signal a bit longer than necessary and then discard samples at the extremities of the ensuing IMFs and residue to get rid of possible boundary effects. Therefore, the backward signal extrapolation carried out in step 2 does not need to be much involved. It can be simple enough to just capture the tendency of the signal trajectory.

Signal extension backward in time should be made for at least the duration of the initial click. For that, extrapolation schemes based on AR modeling [2, 19] can be used. A simpler solution, which is employed here, consists in mirroring the beginning of the input signal with odd symmetry w.r.t. its first sample.

The EMD in step 3 uses a standard version of the algorithm (see Section 2.1). As for the treatment of envelope boundaries, the solution proposed in [10] is resorted to. More specifically, considering the upper envelope for didactic reasons, a straight line is first fitted to the two consecutive maxima nearest to the end (or beginning) of the signal. Then, an artificial new end (or beginning) point

for the envelope is created at the end (or beginning) of the segment. This new point is taken as the largest value between the own signal and the linearly extrapolated envelope. A similar scheme can be employed to extend the lower envelope. In both cases, no extension of the input signal is carried out, just extrapolation of its lower and upper envelopes toward its boundaries.

The aforementioned procedure surely helps to reduce end effects observed in the IMFs, but do not completely eliminate them. Hence, input signal extrapolation performed in step 2 is still needed.

One of the known issues of the standard EMD is the so-called mode-mixing or intermittence [8], which consists of the split of an apparent single intrinsic mode between two adjacent IMFs. Intrinsic mode segregation is a complex question whose discussion is beyond the scope of this paper. As reported in [20] it depends on parameters such as the relative amplitude of the modes and their proximity in frequency.

As one could anticipate from the above discussion, although a single long pulse would qualify for being an IMF, it is not always true that one of the IMFs produced by EMD of the input signal constitutes alone an adequate pulse estimate. The main reason for that seems to be the overlap between the spectral range of the long pulse and the low-frequency content of the audio signal of interest.

An illustration of the mode-mixing problem in the context of EMD-based pulse estimation is depicted in Figure 2. As can be seen, while the tail of the pulse is well captured by the residue obtained after extracting seven IMFs, adequate representation of the initial faster oscillations only happens if the 7th IMF is added to the residue.

Similar to the strategy employed in [6], a practical way to obtain a useful pulse estimate is to predefine three temporal regions for the pulse and assign to each region pulse estimates with different degrees of detail (or frequency ranges). One pulse partition that typically works in practice is the following:

- p_F : about half oscillation cycle from the beginning of the pulse;
- p_M : about one and half oscillation cycles from the end of p_F ;
- p_L : the rest of the pulse from the end of p_M .

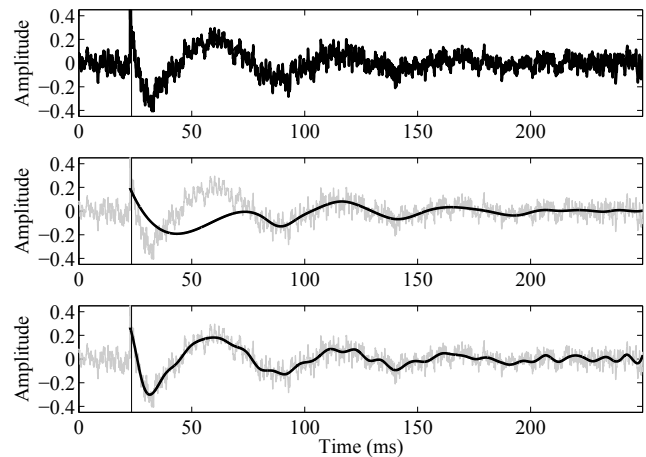


Figure 2: *Top*: Signal corrupted with a long pulse. The thin vertical line at about 25 ms indicates the beginning of the pulse. *Middle*: corrupted signal (thin faded gray line) and residue after extracting the first 7 IMFs (thick black line). *Bottom*: corrupted signal (thin faded gray line) and the sum of the residue with the 7th IMF (thick black line).

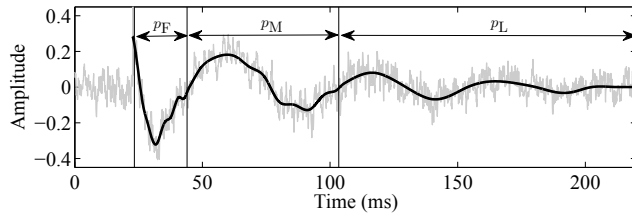


Figure 3: Signal corrupted with a long pulse (thin faded gray line) and composite pulse estimate after EMD with 7 IMFs (thick black line). The thin vertical lines delimit the partitions p_F , p_M , and p_L . The residue is attributed to p_L . The sum of the residue and the 7th IMF is attributed to p_M . The sum of the residue with the 7th and 6th IMFs is attributed to p_F .

The residue of the EMD analysis can be assigned to p_L . Since the maximum number of IMFs can be forcefully limited, the residue is not a monotonic function and could be further decomposed into more IMFs. However, the idea here is to set the maximum number of IMFs so as to produce a residue that, besides being smooth, captures well the oscillations present in the pulse tail p_L . Continuing the decomposition process until obtaining a monotonic residue is then unnecessary and would undesirably increase the computational costs involved.

To the portion p_M one can assign the sum of the residue and the last IMF observed in that region. In a similar fashion, to the portion p_F one can assign the sum of the residue and the two last IMFs observed in that region. In practice, the partitions p_F , p_M , and p_L must overlap a bit in time. This way, it is possible to merge their waveforms together seamlessly via straightforward cross-fading schemes. An example of the proposed partition and assignment scheme is seen in Figure 3, where the cross-fading among adjacent partitions lasts about 4.5 ms.

As reported in [18], EMD of white noise tends to produce IMFs that could be considered as sub-band signals of a dyadic octave filterbank analysis with poor selectivity channels. Thus, the higher the IMF number the lowest the mean frequency of its power spectral density. Assuming this behavior holds for spectrally rich audio signals, one may speculate that the estimates assigned p_L , p_M , and p_F would consist of lowpass versions of the input signal with progressively increasing cutoff frequencies.

From the above discussion, an advantage of the EMD is that the resulting IMFs are pulse estimates with different frequency bandwidths that can be readily combined in the partition and assignment scheme. However, from Figure 3 one perceives that, especially in regions p_F and p_M , part of the low-frequency content of the signal is present in the pulse estimate.

A practical solution to gain more control over the smoothness of the pulse estimate is to post-process the partial pulse estimates obtained in each partition prior to their merging. An effective post-processing is the piece-wise polynomial fitting described in [6]. In brief terms, this signal smoothing scheme consists of fitting a low-order polynomial to short-duration frames of the signal of interest, in an overlap-and-add signal segmentation, e.g., Hanning windows with 50% temporal superposition. If the polynomial order is fixed to 2, as adopted in the conducted experiments, the degree of smoothness is solely controlled by the length of the overlapping windows.

Here, a different window length can be chosen for each partition p_L , p_M , and p_F . A rule of thumb is to set the window length to some value (in units of time or number of samples) between a quarter and a half of the smallest period of the pulse oscillation

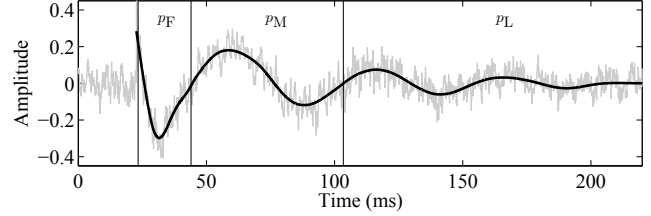


Figure 4: Signal corrupted with a long pulse (thin faded gray line) and composite pulse estimate after applying the piece-wise polynomial smoothing to the estimate shown in Figure 3.

observed in the considered partition. The pulse estimate that results from applying the piece-wise polynomial fitting is seen in Figure 4, where windows of sizes 10 ms, 20 ms, and 30 ms have been used, respectively, to smooth out the pulse estimates in partitions p_F , p_M , and p_L . In order to avoid boundary effects during the smoothing procedure, the bumpy pulse estimate in partition p_F was also extended backward by about 10 ms via odd symmetry reflection w.r.t. its first sample.

Once an adequate pulse estimate is obtained, de-thumping is simply achieved by subtracting the pulse from the signal. Removal of the initial click can be easily accomplished via standard model-based de-clicking techniques [2]. In practice, it may be desirable to artificially overestimate the click duration toward the beginning of the long pulse.

At this stage it seems appropriate to comment on the strong and weak points of the EMD-based pulse estimation. An obvious weakness lies in its inability to obtain a pulse estimate at once as a single IMF. Furthermore, the user is left with the task of choosing several additional parameters for the post-processing stage. This burden, however, can be alleviated through a graphical user interface, similar to that designed and proposed in [6]. On the other hand, the EMD can be seen as a computationally cheap way of obtaining lowpass and bandpass filtered versions of the input signal.

3.2. CEEMD-based Estimation of Single Pulses

CEEMD-based estimation of single pulses follows the first three processing steps defined in the beginning of Section 3.1, the latter carried out with the CEEMD instead of the EMD. The fourth step, however, turns out to be unnecessary, as it will be demonstrated.

Besides the number of IMFs, signal analysis via CEEMD requires the choice of two other parameters: the standard deviation of the additive noise realizations and the number of their pairs. Fortunately, in the context of long pulse estimation tackled here, these two parameters have minor impact to the final results. For all simulations presented in this paper involving CEEMD, the standard deviation of the additive noise has been set to 20% of that of the input signal, whereas the number of noise realization pairs was set to 4, mainly to reduce computational costs.

Once the maximum number of IMFs is defined, the pulse estimate is simply taken as the ensuing residue, after discarding the initial samples that are due to the artificial signal extension. As before, this residue is not a monotonic function and could be further decomposed into more IMFs.

At this point it is worth mentioning that the maximum number of IMFs required for the CEEMD to produce a smooth pulse estimate is about twice as that of EMD. This slower convergence to a target-residue may come from a much higher number of signal extrema in the beginning of the decomposition, due to the addition of noise to the input signal.

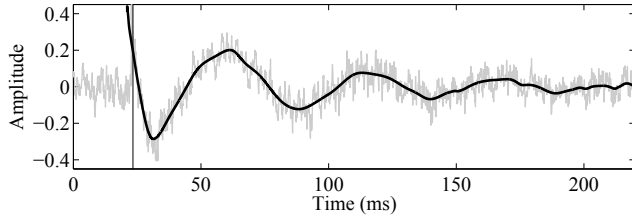


Figure 5: Signal corrupted with a long pulse (thin faded gray line) and pulse estimate (thick black line) as the CEEMD residue after extracting the first 12 IMFs. The thin vertical line at about 23 ms indicates the actual beginning of the pulse. Pulse samples before that limit should be discarded.

Figure 5 displays an example of pulse estimate obtained via the proposed CEEMD-based method, where the pulse is taken as the residue after extracting 12 IMFs. Here, for illustration purposes, the pulse is plotted including the samples related to a backward signal extension of about 2.5 ms. As can be seen, the attained pulse estimate exhibits an adequate level of smoothness and is capable of following the pulse trajectory in both fast and slow oscillation regions.

In comparison with the pulse estimate shown in Figure 4, the one yielded by the CEEMD solution is a bit bumpier. However, since the amplitudes of these faster oscillations are quite small, their subtraction from the corrupted signal is bound to produce inaudible effects. Therefore, post-processing for further smoothing and windowing schemes for pulse composition are no longer needed.

3.2.1. Speeding up the CEEMD-Based Pulse Estimation

As seen in the previous section, the CEEMD-based pulse estimation is effective, yet simpler than the EMD-based counterpart, from the algorithm implementation and calibration perspectives. Its main drawback is a higher computational cost that slows down the process of obtaining the desired pulse estimates.

In the context of EMD of white noise, findings reported in [18] suggest that the number of IMF zero-crossings, which holds relation with the number of IMF extrema, tends to decrease on average by half from a given IMF to the subsequent one. Thus, the larger the number of extrema in the beginning, the longer the decomposition takes to converge to a monotonic residue.

With the previous information in mind, the following modification, which affects the computation of signal envelopes within the EEMD processing chain, has been found operative to speed up the CEEMD-based pulse estimation method:

1. Detect all peaks and valleys of the input signal as usual;
2. Select from the previous set of peaks and valleys only the peak (valley) with maximum (minimum) amplitude inside juxtaposed observation windows of 3.4 ms (about 150 samples at 44.1 kHz sampling rate);
3. Obtain the signal envelopes as usual, but using only the peaks and valleys selected in step 2;
4. Apply item 2 only for extraction of the first two IMFs.

The selection performed in step 2 above is an attempt to retain only the most prominent peaks and valleys of the input signal, which is artificially corrupted with noise in the CEEMD. The proposed peak and valley pruning produces upper and lower envelopes are way smoother than those of the noisy input signal. As

a consequence, the frequency bandwidths of the first two IMFs are larger than those of the corresponding IMFs computed via conventional CEEMD, forcing the modified CEEMD iterations to converge faster to slowly varying IMFs, which are of interest to the present application.

Together with the maximum number of IMFs, the length of the observation window in step 2 above can also be changed by the user as a means to control the degree of smoothness of the pulse estimate. Considering a practical range from 1 to 10 ms, the longer the window length, the smaller the maximum number of extracted IMFs required for the residue to become an adequate pulse estimate.

To illustrate the combined role of the two previously discussed parameters in the final CEEMD-based pulse estimate, outcomes of two different yet equally effective configurations of the CEEMD method are presented in Figure 6 and Figure 7. In connection with these results, Table 1 summarizes the processing parameters adopted in each configuration and the average savings in computational time w.r.t the conventional CEEMD-based pulse estimation.

Visual assessment among the plots shown in Figures 5 to 7 reveals similar pulse estimates. Hence, from Table 1, adoption of the proposed peak (valley) picking for envelope computation within CEEMD is advantageous, for it can produce up to a ten-fold reduction in computational time.

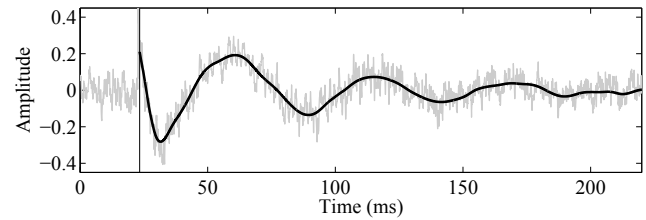


Figure 6: Signal corrupted with a long pulse (thin faded gray line) and pulse estimate (thick black line) as the CEEMD residue after extracting 4 IMFs. Peak and valley selection was carried out within juxtaposed windows of 3.4 ms.

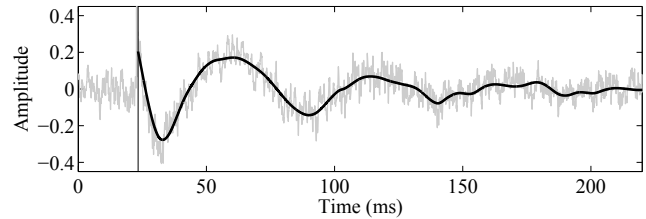


Figure 7: Signal corrupted with a long pulse (thin faded gray line) and pulse estimate (thick black line) as the CEEMD residue after extracting only 2 IMFs. Peak and valley selection was carried out within juxtaposed windows of 6.8 ms.

Table 1: Configuration of the CEEMD-based pulse estimation and corresponding results. The value of T is about 10 s when running the CEEMD analysis on a Core i7 870 2.93 GHz Quad Core CPU. All simulations ran on the same machine.

CEEMD Configuration		Results	
No. IMFs	Window Size	Avg. Proc. Time	Visual Output
13	—	T	Figure 5
4	3.4 ms	$T/3.8$	Figure 6
2	6.8 ms	$T/10$	Figure 7

3.2.2. A Real-World Example of CEEMD-based De-Thumping

As a real-world example, one of the long pulse occurrences in the signal available from [21] has been subjected to the proposed CEEMD-based de-thumping. The signal in question, which is sampled at 22.05 kHz, contains pulses with initial oscillations of about 180 Hz, thus faster than in the previously considered pulse. In order for the CEEMD method to capture those fast pulse variations, the window size in the peak/valley selection scheme has been experimentally set to 1.4 ms, whereas the maximum number of IMFs was limited to 2.

The attained pulse estimate is seen in the top panel of Figure 8, where one can notice undesirable fast oscillations after about 40 ms of the beginning of the pulse. To improve the estimate, they were flattened out via the overlap-and-add polynomial smoothing [6] with windows of 13.6 ms. The final pulse estimate, which is shown in the bottom panel of Figure 8, is then composed by seamlessly merging the first approximately 16 ms of the original CEEMD-based estimate with its smoothed out version from about 40 ms onward.

Concerning the corrupted signal and related pulse estimate depicted in the bottom panel of Figure 8, the corresponding de-thumped version is seen in the top plot of Figure 9. The remaining click of about 680 μ s (about 15 samples at 22.05 kHz sampling rate) was suppressed by LSAR signal reconstruction with model order 65 to produce the signal shown in the middle plot of Figure 9. A detailed vision of the signal reconstruction around the click location is seen in the bottom plot.

3.3. Estimation of Superimposed Pulses

As regards estimation of superimposed pulses, a strategy that has been found effective was to treat independently the parts that form the pulse. In other words, the signal part that follows the last driving initial click is subjected for instance to the CEEMD-based pulse estimator as if it were a single pulse occurrence. Separately, pulse estimation in the intermediate part between two consecutive initial clicks is carried out using the same processing parameters. For this part, it is desirable to extent the signal backward and forward in time for about the duration of the delimiting clicks.

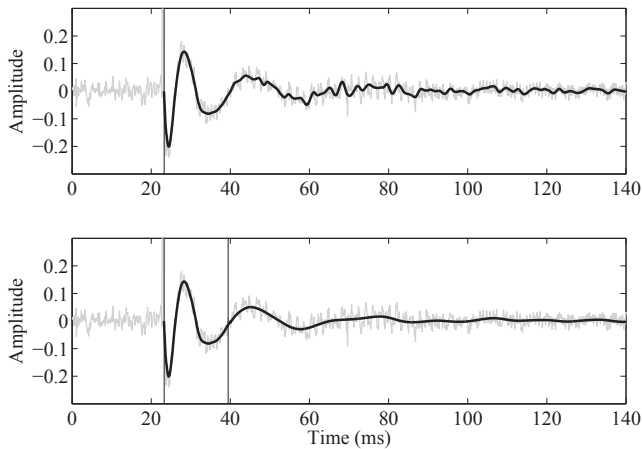


Figure 8: *Top: real-world example of a signal corrupted with a long pulse [21] (faded gray line) and corresponding CEEMD pulse estimate (thick black line). Bottom: same corrupted signal (faded gray line) and improved estimate via polynomial smoothing from 40 ms onward (thick black line).*

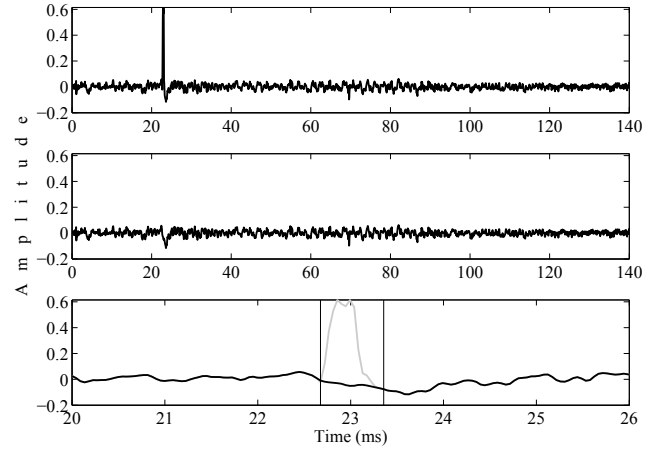


Figure 9: *Top: de-thumped signal related to the corrupted signal and related pulse estimate shown in the bottom plot of Figure 8. Middle: de-clicked signal via LSAR signal reconstruction. Bottom: detail of the signal reconstruction around the click location, with the original click painted in faded gray line. The thin vertical lines around 23 ms delimit the audio region subjected to LSAR interpolation.*

Figure 10 shows an example of CEEMD-based estimation of superimposed pulses in which the same processing setup that generated the result seen in Figure 6 has been used.

4. PERFORMANCE ASSESSMENT

In this paper the performance assessment methodology for audio de-thumping methods defined in [6] is employed. The same set of test signals and one of the quantitative metrics considered in [6] are also used as a means to allow direct comparisons with those previous results. A brief overview of the experimental setup is given in the sequel. The reader is referred to [6] for a more detailed description.

4.1. Test Signals

The test signals comprise a set of reference uncorrupted signals and a corresponding set of artificially corrupted versions. The reference set is composed of 6 CD-quality short-duration (11 to 20 s) excerpts of audio including diverse musical genres such as pop, jazz, classic, and ethnic, as well as solo of drums and acoustic bass.

The corrupted set was produced by adding several single long pulses (with initial click) to the reference signals. Successive pulses were placed approximately 769 ms apart from each other. Here, only the set of strong pulses [6, 22] will be considered for performance evaluation.

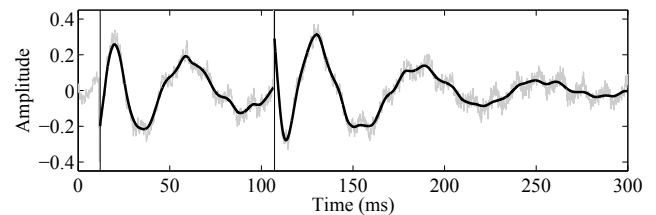


Figure 10: *Signal corrupted with a superimposed long pulse (thin faded gray line) and CEEMD-based pulse estimates (thick black lines).*

4.2. Experimental Setup and Performance Metric

As in [6], the evaluation methodology adopted here consists in first obtaining restored versions (de-thumped and de-clicked) of the corrupted set via a selection of de-thumping methods with pre-defined configurations. Then, the reference and restored sets are compared by objective means.

Here, the Perceptual Audio Quality Measure (PAQM) [11] is used as the performance metric. In a nutshell, the PAQM compares a processed signal w.r.t. a reference and outputs a dissimilarity index that takes into account several properties of the human auditory system, such as masking in time and frequency. The closer to zero the PAQM, the more similar perceptually are the processed and reference signals.

The aim here is to run a direct comparison among the PAQM values associated with the restored signals produced by the following de-thumping methods: AR Separation (ARS), TPSW-based (TPSW), EMD-based pulse estimation (EMD), and CEEMD-based pulse estimation (CEEMD). Therefore, only the processing setup of the EMD and the CEEMD will be defined. The processing parameters employed in the ARS and the TPSW can be found in [6].

4.2.1. EMD Configuration

The following configuration was employed to obtain the EMD-related results:

- Backward signal extension: 100 samples with odd symmetry w.r.t. the first sample;
- Maximum number of IMFs: 6;
- Envelope computation: piecewise cubic Hermite interpolating polynomials across local maxima (minima);
- p_F : 44 ms from the beginning of the pulse;
- p_M : 60 ms from the end of p_F ;
- p_L : 100 ms from the end of p_M ;
- Pulse estimate inside p_F : (residue + 6th IMF + 5th IMF) smoothed out via the overlap-and-add polynomial scheme with windows of 10 ms and 2nd-order polynomials;
- Pulse estimate inside p_M : (residue + 6th IMF) smoothed out via the overlap-and-add polynomial scheme with windows of 20 ms and 2nd-order polynomials;
- Pulse estimate inside p_L : (residue) smoothed out via the overlap-and-add polynomial scheme with windows of 30 ms and 2nd-order polynomials;
- Click removal: 75th-order LSAR signal reconstruction of 50 samples from the click onset.

4.2.2. CEEMD Configuration

The following configuration was employed to obtain the CEEMD-related results:

- Number of noise realization pairs: $N/2 = 4$;
- Standard deviation of each realization: $\varepsilon = 0.2 \text{ std}\{x(t)\}$;
- Backward signal extension: 100 samples with odd symmetry w.r.t. the first sample;
- Maximum number of IMFs: 4;

- Envelope computation (for first two IMFs): piecewise cubic Hermite interpolating polynomials across local maxima (minima), after selection of one maximum (minimum) per juxtaposed windows of 3.4 ms;
- Envelope computation (third and fourth IMFs): piecewise cubic Hermite interpolating polynomials across local maxima (minima);
- Click removal: 75th-order LSAR signal reconstruction of 50 samples from the click onset.

4.3. Results and Discussion

Table 2 summarizes the attained PAQM of the restored (de-thumped and de-clicked) test signals for each of the considered methods and predefined configurations given in the previous sections and [6].

Analysis of the results suggests a tendency of the EMD to outperform the competing methods. For instance, for all test signals restored by EMD, the associated PAQMs are smaller than those of TPSW. ARS offers the most effective restoration for signals *Drums* and *Singing*, whereas CEEMD is the least successful in these cases. Signal *Ethnic* as restored by CEEMD yields the smallest PAQM. It is worth mentioning that too small PAQM differences may not necessarily imply a noticeable perceptual difference. Audio examples can be found in [23].

In summary, for the considered experimental scenario, the attained PAQM results suggest EMD and CEEMD are effective and competitive tools for audio de-thumping.

Table 2: Comparative performance evaluation of the EMD, CEEMD, TPSW, and ARS de-thumping methods using PAQM. The best results (lowest PAQM) are highlighted.

Test Signal	EMD	CEEMD	TPSW	ARS
Pop	0.024	0.028	0.036	0.046
Jazz	0.012	0.031	0.028	0.061
Classic	0.010	0.023	0.017	0.033
Ethnic	0.032	0.030	0.051	0.048
Drums	0.029	0.097	0.049	0.014
Bass	0.015	0.030	0.031	0.188
Singing	0.092	0.282	0.110	0.066

5. CONCLUSIONS

This paper addressed the problem of long pulse removal from audio signals (de-thumping). Two methods for long pulse estimation based on Huang's Empirical Mode Decomposition (EMD) were proposed. After an overview of the EMD and its related complementary ensemble version (CEEMD), their use as tools to obtain adequate pulse estimates from corrupted audio signals was investigated by means of practical examples.

It was found out experimentally that, possibly due to mode mixing issues afflicting signal analysis via EMD, an intrinsic mode function (IMF) or a non-monotonic residue (after successively extracting a given number IMFs from the original signal) may not serve alone as an adequate pulse estimate. To overcome the problem, the adopted solution [6] was to define three temporal partitions to which pulse estimates with different frequency bandwidths or levels of smoothness were attributed. The final composite pulse estimate was then assembled together by merging seamlessly the estimates from each partition.

As regards signal analysis via the CEEMD, it was discovered that adequate pulse estimates could be obtained at once as a non-monotonic residue after successively extracting a given number IMFs from the original signal. However, as compared with the EMD, about twice the number of initial IMFs needed to be extracted for producing a smooth enough pulse estimate. Other CEEMD parameters such as the variance of the additive noise and the number of ensemble pairs had negligible impact on the final results. As a means to decrease the computational cost of the CEEMD for the studied application, a modified scheme for signal envelope computation was devised: piecewise cubic Hermite interpolating polynomials were fitted across local maxima (minima), after selection of one maximum (minimum) per juxtaposed short-duration windows. Average reductions in computation time up to ten times were reported.

Objective performance evaluation of the proposed EMD- and CEEMD-based methods for audio de-thumping was carried out using the same methodology and test data of [6]. Indirect comparative results, in terms of the Perceptual Audio Quality Measure [11] of the restored versions of the corrupted data, suggest the proposed CEEMD-based method tends to perform as effectively as the competing TPSW-based procedure [6] and outperform the AR separation method [2]. As regards the EMD-based method the observed tendency is of a more favorably performance in comparison with the TPSW-based solution.

6. REFERENCES

- [1] S. V. Vaseghi, *Algorithms for Restoration of Archived Gramophone Recordings*, Ph.D. thesis, Cambridge Univ., UK, 1988.
- [2] S. J. Godsill and P. J. W. Rayner, *Digital Audio Restoration — A Statistical Model Based Approach*, Springer-Verlag, London, UK, 1998.
- [3] S. V. Vaseghi and R. Frayling-Cork, “Restoration of Old Gramophone Recordings,” *J. Audio Eng. Soc.*, vol. 40, no. 10, pp. 791–801, Oct. 1992.
- [4] S. J. Godsill, *The Restoration of Degraded Audio Signals*, Ph.D. thesis, Cambridge Univ., UK, 1993.
- [5] S. J. Godsill and C. H. Tan, “Removal of Low Frequency Transient Noise from Old Recordings Using Model-Based Signal Separation Techniques,” in *Proc. IEEE WASPAA*, 1997.
- [6] P. A. A. Esquef, L. W. P. Biscainho, and V. Välimäki, “An Efficient Algorithm for the Restoration of Audio Signals Corrupted with Low-Frequency Pulses,” *J. Audio Eng. Soc.*, vol. 51, no. 6, pp. 502–517, June 2003.
- [7] W. A. Struzinski and E. D. Lowe, “A Performance Comparison of Four Noise Background Normalization Schemes Proposed for Signal Detection Systems,” *J. Acoust. Soc. Am.*, vol. 76, no. 6, pp. 1738–1742, Dec. 1984.
- [8] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, E. H. Shih, Q. Zheng, C. C. Tung, and H. H. Liu, “The Empirical Mode Decomposition Method and the Hilbert Spectrum for Non-stationary Time Series Analysis,” in *Proc. Roy. Soc. London*, 1998, vol. 454A, pp. 903–995.
- [9] N. E. Huang and Z. Wu, “An Adaptive Data Analysis Method for Nonlinear and Nonstationary Time Series: the Empirical Mode Decomposition and Hilbert Spectral Analysis,” in *Proc. 4th Int. Conf. Wavelet Analysis and Its Applications*, China, 2005.
- [10] Z. Wu and N. E. Huang, “Ensemble Empirical Mode Decomposition: A Noise-Assisted Data Analysis Method,” *Advances in Adaptive Data Analysis*, vol. 1, no. 1, pp. 1–41, 2009.
- [11] J. G. Beerends and J. A. Stemerdink, “A Perceptual Audio Quality Measure Based on a Psychoacoustic Sound Representation,” *J. Audio Eng. Soc.*, vol. 40, no. 12, pp. 963–978, Dec. 1992.
- [12] D. Gabor, “Theory of communication. Part 1: The analysis of information,” *J. of IEEE – Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.
- [13] N. E. Huang, Z. Shen, and S. R. Long, “A new view of nonlinear water waves: The Hilbert Spectrum 1,” *Annual Review of Fluid Mechanics*, vol. 31, no. 1, pp. 417–457, 1999.
- [14] N. E. Huang, M. L. C. Wu, S. R. Long, S. S. P. Shen, W. Qu, P. Gloersen, and K. L. Fan, “A confidence limit for the empirical mode decomposition and Hilbert spectral analysis,” *Proc. R. Soc. of Lond. A*, vol. 459, pp. 2317–2345, 2003.
- [15] G. Rilling, P. Flandrin, and P. Gonçalves, “On empirical mode decomposition and its algorithms,” in *Proc. IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, 2003.
- [16] Q. Chen, N. Huang, S. Riemenschneider, and Y. Xu, “A B-spline approach for empirical mode decompositions,” *Advances in Computational Mathematics*, vol. 24, no. 1, pp. 171–195, 2006.
- [17] M. Dätig and T. Schlurmann, “Performance and Limitations of the Hilbert-Huang Transformation (HHT) with an Application to Irregular Water Waves,” *Ocean Engineering*, vol. 31, no. 14, pp. 1783–1834, Oct. 2004.
- [18] P. Flandrin, G. Rilling, and P. Gonçalves, “Empirical Mode Decomposition as a Filter Bank,” *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 112–114, Feb. 2004.
- [19] P. A. A. Esquef and L. W. P. Biscainho, “An Efficient Model-Based Multirate Method for Reconstruction of Audio Signals Across Long Gaps,” *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1391–1400, July 2006.
- [20] G. Rilling and P. Flandrin, “One or Two frequencies? The Empirical Mode Decomposition Answers,” *IEEE Trans. Signal Processing*, vol. 56, no. 1, pp. 85–95, 2008.
- [21] http://dea.brunel.ac.uk/cmsp/Home_Saeed_Vaseghi/Home.html.
- [22] <http://www.acoustics.hut.fi/publications/papers/jaes-IP/>.
- [23] www.lncc.br/~pesquef/dafx11/.

GENERATION OF NON-REPETITIVE EVERYDAY IMPACT SOUNDS FOR INTERACTIVE APPLICATIONS

Wasim Ahmad, Ahmet M. Kondo

I-Lab, Centre for Vision, Speech and Signal Processing (CVSSP)
University of Surrey, Guildford, GU2 7XH, United Kingdom
{w.ahmad, a.kondo}@surrey.ac.uk

ABSTRACT

The use of high quality sound effects is growing rapidly in multimedia, interactive and virtual reality applications. The common source of audio events in these applications is impact sounds. The sound effects in such environments can be pre-recorded or synthesized in real-time as a result of a physical event. However, one of the biggest problems when using pre-recorded sound effects is the monotonous repetition of these sounds which can be tedious to the listener. In this paper, we present a new algorithm which generates non-repetitive impact sound effects using parameters from the physical interaction. Our approach aims to use audio grains to create finely-controlled synthesized sounds which are based on recordings of impact sounds. The proposed algorithm can also be used in a large set of audio data analysis, representation, and compression applications. A subjective test was carried out to evaluate the perceptual quality of the synthesized sounds.

1. INTRODUCTION

Our environment is full of diverse types of impact sounds such as hitting, collision, bumping, dripping, etc. Such impact sounds are generally produced when two or more objects interact with each other. Pre-recorded versions of these sounds are used to generate such sound effects in interactive and virtual reality applications, in real time and offline productions. This method requires a large set of recordings of impact sounds to cover all possible situations which in turn necessitates a very large memory. One possible way to reduce recordings' size is by grouping them by size, material type, etc., but even then many recordings need to be carried out. For this reason, a small set of recordings of impact sounds is generally played back repetitively, and that can be tedious to the listener. Methods have been proposed to improve the realism of sound effects in games, such as the work of Vachon [1]. However, the repetition of sound effects in interactive applications, particularly in game's audio, remains a big challenge for the researcher and audio designer.

Alternatively, impact sounds can be generated automatically using either physics-based interaction of objects, known as physical models, or by imitating the properties of sound as perceived by the listener, known as spectral models. In recent years a number of such synthesis algorithms have been developed and applied to impact sounds synthesis [2, 3, 4, 5, 6, 7, 8, 9]. Physical models [2, 3, 4, 5] are very efficient and accurate in simulating a target sound but the refinement of such models is not always successful because the physical mechanisms of many environmental impact sounds are still not completely understood [10]. Therefore, a limited class of impact sounds has been targeted by this

type of models. Furthermore, these models are computationally-intensive and require significant parameter-tuning to achieve realistic results, making it more difficult to use in a game production pipeline. In contrast, spectral models [6, 7, 8, 9] have a broader scope and construct the spectrum as received by the ear. Therefore, their refinement and repurposing is easier than physical models.

In recent years, combinations of sound synthesis models with pre-recorded sound have been used to generate high quality impact sound in interactive applications [11, 12]. Such approaches reduce the effect of the monotonous repetition of recorded sounds, and enhance the quality of synthesized sounds by linking the synthesis parameters to the physics engine. Bonneel *et al.* [11] presented a new frequency-domain method that used both pre-recorded sounds and physical models to generate high quality sounds. In [12], Picard *et al.* proposed a technique where non-repetitive sound events can be synthesized for interactive animations by retargeting the audio grains, extracted from the recorded sounds, based to the parameters received from the physics engine.

In this paper, we propose a similar approach where the pre-recorded impact sounds are represented in the form of a dictionary and synthesis patterns. During the generation phase, the synthesis pattern and corresponding atoms from the dictionary are selected according to the reported synthesis parameters from the physical interaction. During the analysis process, a continuous pre-recorded impact sounds are automatically segmented into individual events and all the events collected from different impact sound sources are decomposed into sound grains, where each grain has energy only at a particular frequency or scale. A dictionary is trained from the extracted sound grains. The recorded impact sound events are projected onto the dictionary which constitutes the synthesis patterns. During synthesis process, these patterns are tuned according to the target sound parameters.

2. SIGNAL REPRESENTATION TECHNIQUES

For many years, a large family of signal analysis techniques have heavily relied on Fourier transform (FT) and short-time Fourier transform (STFT) where the input signal is represented with the superposition of fixed basis functions i.e. sinusoids. The FT and STFT methods are most useful when considering stationary signals but most real-world sound signals are not stationary in time. Therefore, these analysis techniques are inadequate for such signals. Over the last two decades there has been a lot of interest to find alternative signal representation techniques which are adaptive and specialized to the signals under consideration. As a result, a number of basis functions and representation techniques have been developed to represent any input signal in a more compact, efficient, and meaningful way.

2.1. Dictionary-Based Methods

One of these techniques, which have attracted a lot of interest in recent years, is dictionary-based representation as it offers a compact form of the signal, and is highly adaptive. These methods have been used in many signal processing applications including analysis and representation of audio signals [13, 14] and music [15].

In dictionary-based methods, a signal is represented as a linear combination of elementary waveforms (atoms) taken from a dictionary. A dictionary is a collection of parameterized waveforms. Let \mathbf{x} be a discrete-time real signal of length N i.e. $\mathbf{x} \in \mathbb{R}^N$, and $\mathbf{D} = [\delta_1, \delta_2, \dots, \delta_K]$ be a dictionary, where each column δ_k represents an atom and its length is N i.e. $\mathbf{D} \in \mathbb{R}^{N \times K}$. The aim is to represent \mathbf{x} as a weighted sum of atoms δ_k which can be written as,

$$\mathbf{x} = \sum_{k=1}^K \delta_k w_k \quad (1)$$

where \mathbf{w} is a column vector in \mathbb{R}^K and represents the expansion coefficients or weights. Generally, the dictionary \mathbf{D} is overcomplete i.e. $N < K$, which means the matrix \mathbf{D} is of rank N and the linear system in Eq. (1) is undetermined. In that case, the decomposition vector \mathbf{w} in Eq. (1) is not unique and there may even be an infinite number of possible expansions of the form of Eq. (1). Therefore, one has to introduce some additional constraints to specify a unique or particular decomposition.

2.2. Sparse Representations

Given an overcomplete dictionary \mathbf{D} and the signal \mathbf{x} , finding a solution to the underdetermined systems given in Eq. (1) is a non-trivial task. In general, the representation in Eq. (1) is approximated by applying some additional constraints to specify a unique or particular solution. An adequate approximation of the signal \mathbf{x} in Eq. (1) is obtained by selecting few atoms δ_k from dictionary \mathbf{D} corresponding to highest weights w_k . That is, useful representations are the ones where most of the energy of the signal \mathbf{x} is concentrated into a small number of coefficients, hence \mathbf{x} can be approximated using only j atoms from the predefined dictionary as

$$\mathbf{x} = \sum_{k=1}^j \delta_k w_k + \mathbf{r} \quad (2)$$

or in matrix form

$$\mathbf{x} = \mathbf{D}\mathbf{w} + \mathbf{r} \quad (3)$$

where $j < K$ and $\mathbf{r} \in \mathbb{R}^N$ is residual. The selection of atoms and their numbers are controlled by limiting the value of approximation error. By applying such criterion, the approximation solution given in Eq. (3) can be redefined as

$$\mathbf{x} \approx \mathbf{D}\mathbf{w} \text{ such that } \|\mathbf{x} - \mathbf{D}\mathbf{w}\|_2 \leq \epsilon \quad (4)$$

where ϵ is a given small positive number. The solution with the fewer number of atoms and corresponding weights is certainly an appealing representation. Sparse or compact approximation of a signal \mathbf{x} is measured using the ℓ_0 criterion, which counts the number of non-zero entries of the weights vector $\mathbf{w} \in \mathbb{R}^K$. The problem of finding the optimally sparse representation can be defined as the solution to

$$\min_{\mathbf{w}} \|\mathbf{w}\|_0 \text{ such that } \|\mathbf{x} - \mathbf{D}\mathbf{w}\|_2 \leq \epsilon \quad (5)$$

where $\|\mathbf{w}\|_0$ is the ℓ_0 -norm, which count the number of non-zero coefficients in weight vector \mathbf{w} . The problem of finding the optimally sparse representation, i.e., with minimum $\|\mathbf{w}\|_0$, is a combinatorial optimization problem in general. Constraining the solution \mathbf{w} to have the minimum number of nonzero elements creates an NP-hard problem [16] and cannot be solved easily. Therefore, approximation algorithms, such as matching pursuit (MP) [17], orthogonal matching pursuit (OMP) [18], and basis pursuit (BP) [19], are used to find an optimal approximation solution of Eq. (5). The MP and OMP algorithms are classified as greedy methods where a signal approximation is iteratively built up by selecting the atom that maximally improves the representation at each iteration. These algorithms converge rapidly, and exhibit good approximation properties for a given criterion [17, 20].

2.3. Selection of Dictionary

Dictionaries are often constructed from a combination of discretized, scaled, translated, and modulated lowpass functions. An overcomplete dictionary that leads to sparse representations can either be chosen as a prespecified set of functions or designed by adapting its content to fit a given set of signal examples. Choosing a prespecified transform matrix is appealing because it is simpler but there is no guarantee that these bases will lead to a sparse representation of signals under consideration.

The sparse approximation of the Eq. (5) can also be improved by using an appropriate dictionary for the given class of signals. Instead of using predetermined dictionaries, dictionary learning methods [21, 15] can be used to refine them. Such methods adapt an initial dictionary to a set of training samples. Therefore, the aim is to learn a dictionary for which an input signal, taken from a given class of signals, has a sparse approximation.

3. PROPOSED ANALYSIS-SYNTHESIS ALGORITHM

The proposed synthesis algorithm generates the target impact sounds using parametric representation modeled from the recorded impact sounds. This algorithm is divided into three stages i.e. analysis, parameterization, and synthesis, as depicted in Fig. 1. In the analysis phase, the recorded continuous impact sounds are segmented and split into sound grains. During the parameterization phase, the impact sounds are represented by synthesis patterns, and an adaptive dictionary trained from these sound grains. The target sound is generated at the synthesis stage where a pattern is selected and adjusted according to the parameters received from the physical interaction.

4. ANALYSIS OF RECORDED SOUNDS

The aim of the analysis process is to extract the sound grains which characterize the recorded impact sounds. The analysis stage includes the segmentation, peak alignment, and the extraction of sound grains.

4.1. Automatic Segmentation

The first step during the off-line analysis of the impact sound is to segment each recorded sound signal into individual *sound events* or simply *events*. For example, if the input sound is a clapping sound then each clap in the sound sequence is called an *event*,

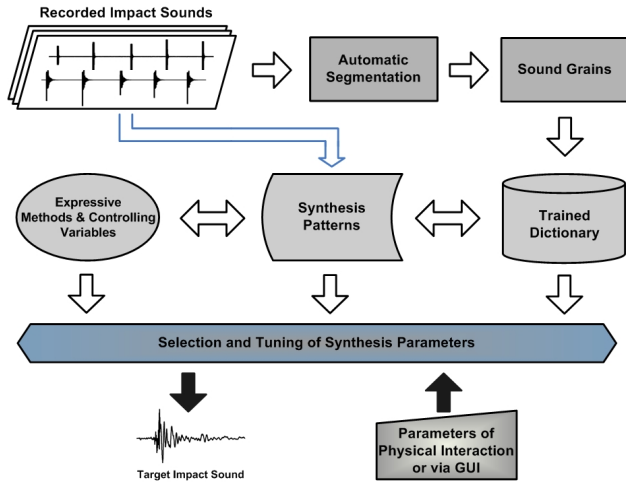


Figure 1: Overview of the proposed analysis-synthesis algorithm.

which is represented by \mathbf{x} . Each event is isolated by detecting and labeling its onset and offset points.

An impulsive event consists of an attack and a decay parts which are concatenated together. The onset of events is labeled using the energy distribution method proposed by Masri *et al.* [22], which detects the beginning of an impulsive event by observing the suddenness and the increase in energy of the attack transient. Detection is selected as an onset of an event when there is a significant rise in energy along with an increased bias towards the higher frequencies. Short-time energy of the signal is used to locate the offset of each event. Starting from the onset of each event, the short-time energy is calculated with overlapped frames, and compared against a constant threshold to determine the offset. Onset detection methods have been applied to input sounds that are monophonic i.e. only a single melodic line or tune is present, and music notes or events do not overlap [23, 24]. In this paper, we have also assumed that there is no overlapping of the sources in the recorded impact sounds.

Equal or different number of events can be selected from each sound source. Once the events are selected and segmented, they are peak aligned by cross-correlation such that the highest peaks occur at the same point in time. This increases the similarities between the extracted sound grains and improves the dictionary learning process. The set of collected sound events can be represented as a matrix \mathbf{X} i.e.,

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m] \quad (6)$$

where each column represents a sound event and length of each event is n . Zero padding is used for any segmented sound event whose length is less than n .

4.2. Extraction of Sound Grains

The recorded impact sounds from different sources need to be represented in a way that i) the similarities and differences between various impact sounds can be observed and parameterized, ii) this parametric representation can be manipulated in various ways to generate sound effects at synthesis stage. Impact sound belongs to the transient signal family that is non-stationary. Based on the frequency resolution properties of the human auditory system, such

signals can be split into layers of grains where the energy of each grain is presented at a particular frequency or scale. The information in each grain and the overall structure of these grains are analyzed based on human auditory system. Such parametric representation can be used to compare the characteristics of different sounds [25]. Furthermore, during the synthesis process, the parameters representing these grains can be manipulated in various ways to control the generated sound.

In the proposed scheme, stationary wavelet transforms (SWT) [26, 27] is used to extract the sound grains from the impact sound events. The SWT is the real-valued extension to the standard discrete wavelet transform (DWT). SWT is preferred over DWT because the latter lacks the property of shift-invariance. The SWT has the ability to underline and represent time-varying spectral properties of the transient signals and offers localization both in time and frequency.

The SWT is applied to each event, \mathbf{x}_i , which decomposes it into two sets of wavelet coefficient vectors: the approximation coefficients \mathbf{ca}_1 and the detail coefficients \mathbf{cd}_1 , where the subscript represents the level of decomposition. The approximation coefficients vector \mathbf{ca}_1 is further split into two parts, \mathbf{ca}_2 and \mathbf{cd}_2 , using the scheme shown in Fig. 2(a). This decomposition process continues up to L^{th} level which produces the following set of coefficient vectors: $[\mathbf{cd}_1, \mathbf{cd}_2, \dots, \mathbf{cd}_L, \mathbf{ca}_L]$. The approximation coefficients represent the low-frequency components, whereas the detail coefficients represent the high-frequency components. To construct the sound grains from coefficients vectors, the inverse SWT is applied to each coefficient vector individually by setting all others to zero which produces the following bandlimited sound grains: $[\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_{L+1}]$. Each grain contains unique information from the sound event, retains the size of the the sound event. The block diagram of the process of extraction of sound grains from a coefficient vector is shown in Fig. 2(c). The entire sound event matrix \mathbf{X} is split into sound grains which produce the grain matrix $\mathbf{G} = [\mathbf{g}_i : i = 1, 2, \dots, p]$, where \mathbf{g}_i form the columns of the grain matrix and the number of total grains are $p = m \times (L + 1)$.

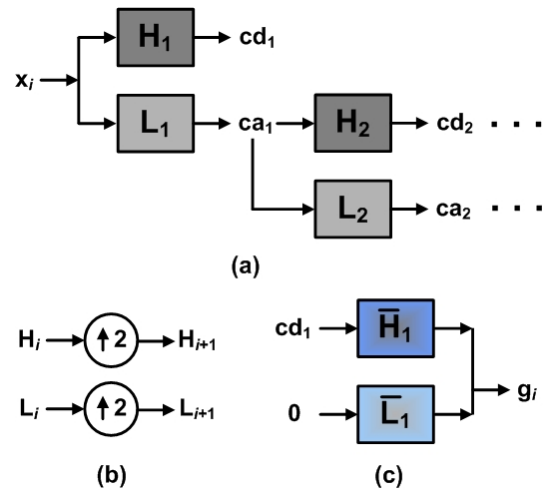


Figure 2: (a) Decomposition tree of SWT, (b) SWT filters, (c) construction of a sound grain.

The selection of wavelet type from the family of wavelets (i.e. Haar, Daubechies, etc.) and their decomposition level depend on

the input sound signal, application area, and the representation model. This is an iterative process where the best wavelet type and optimum decomposition level are obtained by evaluating the perceived quality of the synthesized sounds generated from the different wavelet types and decomposition levels.

5. PARAMETERIZATION

The proper parameterization of the sound features extracted from the analysis part is an essential element of the synthesis systems. In this paper, a dictionary-based approach is used to create a parametric representation of the recorded sounds. The similarities and differences of the sound grains, as well as their relationships to the input sounds are preserved and reflected in the presented parametric representation. One key advantage of dictionary-based signal representation methods is the adaptivity of the composing atoms. This gives the user the ability to make a decomposition suited to specific structures in a signal. Therefore, one can select a dictionary either from a pre-specified set of bases functions, such as wavelets, wavelet packets, Gabor, cosine packets, chirplets, warplets etc., or design one by adapting its content to fit a given set of signals, such as dictionary of instrument-specific harmonic atoms [15].

5.1. Dictionary Learning

Choosing a set of prespecified basis functions is appealing because of its simplicity but there is no guarantee that these basis functions will lead to a compact representation of given signals. The success of such dictionaries in practice depends on how suitable they are to sparsely describe the signals in question. However, there are many potential application areas, such as transient and complex music sound signals, where fixed basis expansions are not well suited to model this type of sound signals. A compact decomposition is best achieved when the elements of the dictionary have strong similarities with the signal under consideration. In this case, a fewer set of more specialized basis functions in the dictionary is needed to describe the significant characteristics of the signal [15, 28, 29]. Ideally, the basis itself should be adapted to the specific class of signals which are used to compose the original signal. As we are dealing with a specific class of sound signals, we believe that it is more appropriate to consider designing learning-based dictionaries.

Given training impact sounds and using adaptive training process, we seek a dictionary that yields compact representations of the sound event matrix \mathbf{X} . The K-SVD algorithm [21] is such a technique for training a dictionary from given example signals. It is a highly effective method, and has been successfully applied to several image processing tasks [30, 31]. The K-SVD algorithm consists of an iterative process of optimization to produce a sparse representation of the given samples based on the current dictionary, and an update of the atoms that best represent the samples. The update of the dictionary columns is done along with an update of the sparse representation coefficients related to it, resulting in accelerated convergence.

In the proposed scheme, the K-SVD algorithm is used to train an adaptive dictionary \mathbf{D} which determines the best possible representation of a given impact sounds. The K-SVD algorithm takes the sound grains matrix \mathbf{G} , as initial dictionary \mathbf{D}_0 , a number of iterations j , and a set of training signals, i.e sound event matrix \mathbf{X} . The algorithm aims to iteratively improve the dictionary to achieve sparser representations of the sound events in \mathbf{X} , by solving the

optimization problem

$$\min_{\mathbf{w}_i} \|\mathbf{x}_i - \mathbf{D}\mathbf{w}_i\|_2^2 \text{ such that } \forall i \|\mathbf{w}_i\|_0 \leq T_0 \quad (7)$$

where T_0 is the number of non-zero entries in \mathbf{w}_i . The iteration of K-SVD algorithms is performed in two basic steps: i) given the current dictionary, the sound events in \mathbf{X} are sparse-coded which produce the sparse representations matrix \mathbf{W} , and ii) using this current sparse representations, the dictionary atoms are updated. The dictionary update is performed one atom at a time, optimizing the target function for each atom individually while keeping the other atoms fixed.

5.2. Synthesis Pattern

The OMP is used to find the synthesis patterns of the input impact sound events over the dictionary. The OMP is a greedy step-wise regression algorithm. The aim of OMP algorithm is to approximate the solution of the sparsity-constrained sparse coding problem given in Eq. (7), where the dictionary atoms have been normalized. At each stage, this algorithm selects the dictionary atom with the maximal projection onto the residual signal. Once the atom is selected, the signal is orthogonally projected to the span of the selected atoms, the residual is recomputed, and the process is repeated. The algorithm stops after a predetermined number of steps, selecting a fixed number of atoms T_0 in every iteration. At this stage, the impact sound matrix \mathbf{X} can be fully represented as a dictionary matrix \mathbf{D} and synthesis patterns matrix \mathbf{W} . The information about the impact sound sources is labeled onto synthesis pattern \mathbf{W} for future reference and for possible use during the synthesis process.

6. GENERATION OF TARGET SOUND

To synthesize the target impact sound, the controlling variables are employed to select the best sound parameters. During the synthesis process, an impact sound event from the sound matrix \mathbf{X} is synthesized by selecting the decomposition pattern \mathbf{w}_i and then adding the corresponding weighted dictionary atoms, which can be written as,

$$\hat{\mathbf{x}}_i \cong \sum_{j \in J} \delta_j \mathbf{w}_i(j) \quad (8)$$

where J contains the T_0 number of indices of the non-zero entries in \mathbf{w}_i . The perceptual quality of the synthesized impact sound event $\hat{\mathbf{x}}_i$ is directly related to the number of non-zero entries in \mathbf{w}_i . The quality of synthesized impact sound event $\hat{\mathbf{x}}_i$ improves sharply for the first few atoms but become imperceptible after a particular value of T_0 .

6.1. Expressive Synthesis Method

Two sound events generated consecutively by the same sound source will be similar but not identical. For example, when a person claps twice in the same way with the same applied force, the generated clapping sounds will be similar but not identical. The proposed algorithm can synthesize example impact sounds approximately from the represented parameters, i.e. synthesis pattern \mathbf{W} and dictionary atoms \mathbf{D} . A limited number of impact sound events sequence can be generated from this representation as the number of synthesis pattern vectors is limited and fixed. Therefore, the same set of impact sounds will be repeated during long impact

sound sequences, which will make it perceptually artificial in the ears of the listeners.

To generate more natural and customized sounds, the proposed method modifies the synthesis process given in Eq. (8). This equation uses the represented parameters, \mathbf{W} and \mathbf{D} , to synthesize an impact sound event. Every time Eq. (8) is executed to synthesize an impact sound event $\hat{\mathbf{x}}_i$, a synthesis pattern \mathbf{w}_i is used to combine the dictionary atoms. For expressive synthesis, when an impact sound event $\hat{\mathbf{x}}_i$ is generated, a small random vector ψ is added to the selected synthesis pattern \mathbf{w}_i such that the overall time-varying spectrum of the impact sound is unchanged. The value of ψ is generated randomly in a sphere of radius R with the origin at the synthesis pattern of the generated impact sound. A different vector ψ is generated for every event of impact sound and the length of ψ is equal to T_0 because only non-zero entries in \mathbf{w}_i are changed. Hence, The synthesis equation given in Eq. (8) is modified for the expressive synthesis process and can be rewritten as,

$$\hat{\mathbf{x}}_i \cong \sum_{j \in J} \delta_j [\mathbf{w}_i + \psi](j). \quad (9)$$

The impact sound sequence generated using Eq. (9) will be similar but not identical, and they will also not be exact copies of the sound events matrix \mathbf{X} .

7. SUBJECTIVE EVALUATION OF SYNTHESIS SOUND QUALITY

Subjective tests have been used to accurately assess the quality of the sound events generated by the proposed algorithm.

7.1. Impact Sound Database

A sample of commonly heard everyday impact sounds were used to evaluate the perceptual quality of sounds synthesized using the proposed analysis-synthesis algorithm. The group contains six impact sounds which include: bumping sounds of a tennis ball, a football and a basketball on laminate floor; a finger knocking sound on a wooden table; and male and female clapping sounds. The recordings of these sounds were made in an acoustical booth ($T_{60} < 100$ ms) at a sampling rate of 44.1 kHz.

To record the bumping sounds, each ball was dropped on laminated floor from a fixed height of 80 cm with no applied force. After each bump, the ball is lifted up to the same height and dropped again. A microphone was placed vertically close to the floor level and horizontally about 100 cm away from the potential point of contact at the floor. The experimenter knocked the centre of the wooden table¹ top with his right hand index finger with a constant force. To capture this sound, the microphone was placed at the level of table top and about 100 cm away horizontally from the centre of the table. The recording of the clapping sounds was made with one male and one female subjects², both between the age of 25 and 35. Each subject was seated in the acoustical booth alone and a microphone was placed about 100 cm away from their hands. Subjects were asked to clap at their most comfortable or natural rate using his or her conventional clapping style. A set of sequences was recorded for each sound source where each sequence contains series of event.

¹The size of the table was 20.5 cm width, 39.5 cm length, and 28.5 cm height.

²The male clapper was the author and the female clapper was a research fellow at I-Lab.

7.2. Synthesis Model and Stimuli

The purpose of this listening test is to compare the quality of the synthesized impact sounds with the original recorded sounds. The set of sequences of six impact sounds from the recorded database were segmented into individual sound events. An equal number of sound events, i.e. 30, were taken from each impact sound type. The segmented sound events were peak aligned and put into a matrix form i.e. $\mathbf{S} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m]$, where $m = 6 \times 30 = 180$ was the number of collected events and the length of each event was $n = 2048$. To extract the sound grains from the collected event matrix \mathbf{X} , the SWT was applied to each event up to the 5th level. That produced the sound grains matrix $\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_p]$, where $p = 180 \times 6 = 1080$ represented the number of sound grains and the length of each grain was equal to $n = 2048$. The parametric representation of the input impact sounds was done by training an adaptive dictionary using the extracted sound grains. Given the sound grains matrix \mathbf{G} as an initial dictionary \mathbf{D}_0 , and the impact sound events matrix \mathbf{X} as training signals, K-SVD algorithm was used to train a final adaptive dictionary $\mathbf{D} = [\delta_1, \delta_2, \dots, \delta_K]$, with a number of atoms equal to $K = 108$. To find the decomposition patterns \mathbf{W} , OMP was used to project recorded sound events over the dictionary \mathbf{D} . Hence, the decomposition patterns \mathbf{W} and adaptive dictionary \mathbf{D} fully described the parametric representation of the input impact sounds.

Three groups of stimuli were synthesized from the represented model of the recorded sounds \mathbf{W} and \mathbf{D} . The first group of stimuli was synthesized using seven atoms from the represented model, while in the second and third groups, they were synthesized using fourteen and twenty one atoms respectively from the represented model. Furthermore, each group contains twelve stimuli, where two stimuli are used from each sound source: male clapper, female clapper, tennis ball, football, basketball, and one finger and table. During the synthesis process, one event of the target sound was generated from the represented model using seven, fourteen, and twenty one atoms. However, when this event was used as a stimulus, the same event was repeated three times with 0.5 seconds interval. Similarly, the corresponding reference sound (the original recorded event) was also repeated three times with 0.5 seconds interval. During each experiment, one reference stimulus and a corresponding synthesized stimulus from each group were presented to the subjects simultaneously. The subjects listened to the reference and synthesized stimuli and graded the quality of the synthesized sounds. The subjects' responses were collected using the graphical user interface (GUI).

7.3. Subjects and Evaluation Setup

A group of 10 subjects (8 male and 2 female), between the age of 26 and 40, participated in the subjective evaluation. The subjects included PhD students and staff from the I-Lab centre with no reported hearing impairment. The subjects were trained before the evaluation session and can be considered to be expert listeners.

For the evaluation experiment, the subjects were seated in an isolated multimedia room. The experimental setup consisted of one Dell Inspiron 630m laptop and one Sennheiser HD 500 headphone. Every subject was familiarized with the evaluation process by undertaking a training session. A GUI was built in MATLAB which was used for the training and sound quality evaluation processes.

During each experiment, the subjects were presented with one reference stimulus (the original recording) and three test stimuli

(one for each group) from the same sound source. Since there were twelve experiments, the total number of synthesized sounds evaluated by each subject was equal to 36. The subjects' task was to listen to the reference and the three test sounds, and then rate the quality of the test sounds in comparison to the reference stimulus. The subjects can replay the reference and test sounds as many times as they wished. To register a rating for each test, the subjects were asked to move the slider on a scale ranging from 0 to 100. The 0 to 100 scale is divided into five equal quality steps: Excellent (81-100), Good (61-80), Fair (41-60), Poor (21-40), and Bad (0-20). Once subjects completed all test sounds within a particular experiment, they could move to the next one by clicking the "save and proceed" button, which stored the rating and presented them with the following set of tests. Each subject took about 15 minutes to evaluate all the experiments.

7.4. Results

Fig. 3 shows the mean evaluation ratings from all the subjects as well as the bars 95% confidence intervals of the mean ratings. It can be observed that the higher the number of atoms used in the synthesis process, the better the perceived quality. Furthermore, the relationship between the perceived quality of the synthesised sound and the number of atoms is linear. This result is due to the fact that as the number of atoms increases in the synthesized pattern, the synthesized sound event approximate more closely to the original signal. The figure also shows that even with a small number of atoms, $T_0 = 7$ out of the size of the dictionary $K = 108$, the mean subjects' rating of the quality was "Good". This indicates that our method achieved a perceptually acceptable level of sound quality with only few number of atoms, hence a more compact form. When increasing the number of atoms to $T_0 = 21$, the mean quality rating improved to "Excellent". These results highlight the efficiency of the parameterization technique used, and the advantages of using an adaptive dictionary trained from sound grains that are extracted from the input signal.

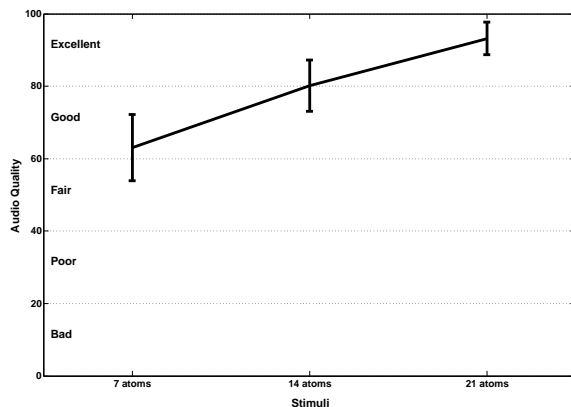


Figure 3: Mean synthesis quality of the synthesized sounds averaged across all subjects. The bars show 95% confidence intervals of the mean ratings.

8. CONCLUSIONS

We presented a new algorithm, which can synthesize any impact sound by analyzing and representing the recorded sound as a set of atoms and synthesis patterns. The atoms of the dictionary were first adaptively trained from the input sound using K-SVD algorithm, and then the synthesis patterns were generated by projecting the sound events over the trained dictionary. The target sound was synthesized by selecting and tuning the synthesis pattern and their corresponding atoms from the dictionary. In addition, an expressive synthesis method was presented which can generate non-repetitive and customized impact sounds. Subjective tests were carried out to evaluate the perceptual quality of the synthesis model. The tests' results showed that it is possible to achieve a satisfactory level of perceived sound quality using the compact representation of a given impact sound with a small number of atoms ($T_0 = 7$) from the trained dictionary. An approximation sound with $T_0 = 21$ was sufficient to yield an "Excellent" quality average rating.

As part of future work, we will further investigate the expressive synthesis model and analyze the distribution of the synthesis patterns of real life sound events and their possible statistical modeling. The quality and realism of the synthesized impact sounds generated from expressive synthesis model will be evaluated using subjective tests.

9. REFERENCES

- [1] Jean-Frederic Vachon, "Avoiding tedium: Fighting repetition in game audio," in *Proc. of Int. Audio Engineering Society Conference: Audio for Games*, London, UK, Feb 2009.
- [2] Kees Van Den Doel, P G Kry, and D K Pai, "Foleyautomatic: Physically-based sound effects for interactive simulation and animation," in *Proc. of ACM Int. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH-01)*, Los Angeles, USA, Aug 2001, pp. 537-544.
- [3] Matthias Rath, Davide Rocchesso, and Federico Avanzini, "Physically based real-time modeling of contact sounds," in *Proc. of Int. Computer Music Conf. (ICMC-2002)*, Goteborg, Sweden, Sep 2002.
- [4] Leevi Peltola, Cumhur Erkut, Perry R Cook, and Vesa Välimäki, "Synthesis of hand clapping sounds," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 3, pp. 1021-1029, Mar 2007.
- [5] Niels Böttcher and Stefania Serafin, "Design and evaluation of physically inspired models of sound effects in computer games," in *Proc. of Int. Audio Engineering Society Conference: Audio for Games*, London, UK, Feb 2009.
- [6] Mitsuko Aramaki and Richard Kronland-Martinet, "Analysis-synthesis of impact sounds by real-time dynamic filtering," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 2, pp. 695-705, Mar 2006.
- [7] Ananya Misra, Perry R Cook, and Ge Wang, "Musical tapestries: Re-composing natural sounds," in *Proc. of Int. Computer Music Conf. (ICMC-06)*, New Orleans, U.S, Nov 2006.
- [8] Wasim Ahmad, Huseyin Hacibiboğlu, and Ahmet M. Konzo, "Analysis-synthesis model for transient impact sounds

- by stationary wavelet transform and singular value decomposition,” in *Proc. of Int. Computer Music Conf. (ICMC-08)*, Belfast, Northern Ireland, Aug 2008, pp. 49–56.
- [9] Wasim Ahmad, Huseyion Hacıhabiboğlu, and Ahmet M. Kondo, “Morphing of transient sounds based on shift-invariant discrete wavelet transform and singular value decomposition,” in *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-09)*, Taipei, Taiwan, Apr 2009, pp. 297–300.
- [10] Perry R. Cook, “Physically informed sonic modeling (phism): Synthesis of percussive sounds,” *Computer Music Journal*, vol. 21, no. 3, pp. pp. 38–49, Autumn 1997.
- [11] Nicolas Bonneel, George Drettakis, Nicolas Tsingos, Isabelle Viaud-Delmon, and Doug James, “Fast modal sounds with scalable frequency-domain synthesis,” *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, vol. 27, no. 3, August 2008.
- [12] Cécile Picard, Nicolas Tsingos, and Francois Faure, “Retargeting example sounds to interactive physics-driven animations,” in *Proc. of Int. Audio Engineering Society Conference: Audio for Games*, London, UK, Feb 2009.
- [13] Remi Gribonval and Emmanuel Bacry, “Harmonic decomposition of audio signals with matching pursuits,” *IEEE Transactions on Signal Processing*, vol. 51, no. 1, pp. 101–111, Jan 2003.
- [14] Bob L. Sturm, Curtis Roads, Aaron McLeran, and John J. Shynk, “Analysis, visualization, and transformation of audio signals using dictionary-based methods,” in *Proc. of Int. Computer Music Conf. (ICMC-2008)*, Belfast, Northern Ireland, Aug 2008.
- [15] P. Leveau, E. Vincent, G. Richard, and L. Daudet, “Instrument-specific harmonic atoms for mid-level music representation,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 1, pp. 116–128, Jan 2008.
- [16] B. K. Natarajan, “Sparse approximate solutions to linear systems,” *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, Apr 1995.
- [17] Stephane G. Mallat, “Matching pursuits with time-frequency dictionaries,” *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec 1993.
- [18] Y. C. Pati, R. Rezaifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition,” in *Proc. of Asilomar Conf. on Signals, Systems and Computers*, Nov 1993, vol. 1, pp. 40–44.
- [19] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, Dec 1998.
- [20] Laurent Daudet, “Audio sparse decompositions in parallel : Let the greed be shared,” *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 90–96, Mar 2010.
- [21] Michal Aharon, Michael Elad, and Alfred Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov 2006.
- [22] Paul Masri and Andrew Bateman, “Improved modeling of attack transients in music analysis-resynthesis,” in *Proc. of Int. Computer Music Conf. (ICMC-96)*, Hong Kong, China, Aug 1996, pp. 100–103.
- [23] Xavier Rodet and Florent JallietRodet:2001, “Detection and modeling of fast attack transients,” in *Proc. of Int. Computer Music Conf. (ICMC-01)*, Havana, Cuba, 2001, pp. 1–4.
- [24] Juan P. Bello and Mark Sandler, “Phase-based note onset detection for music signals,” in *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-03)*, Apr 2003, vol. 5, pp. 441–444.
- [25] Simon Tucker and Guy J. Brown, “Classification of transient sonar sounds using perceptually motivated features,” *IEEE Journal of Oceanic Engineering*, vol. 30, no. 3, pp. 588–600, Jul 2005.
- [26] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, “A real-time algorithm for signal analysis with the help of the wavelet transform,” in *Wavelets: Time-Frequency Methods and Phase Space*, Jean-Michel Combes, Alexander Grossmann, and Philippe Tchamitchian, Eds. 1989, pp. 289–297, Springer-Verlag.
- [27] G. P. Nason and B.W. Silverman, “The stationary wavelet transform and some statistical applications,” in *Lecture Notes in Statistics*, 103, pp. 281–299. 1995.
- [28] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, “Sparse representations in audio and music: From coding to source separation,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995–1005, Jun 2010.
- [29] Michael S. Lewicki, Terrence J. Sejnowski, and Howard Hughes, “Learning overcomplete representations,” *Neural Computation*, vol. 12, no. 2, pp. 337–365, Feb 2000.
- [30] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, Dec 2006.
- [31] J. Mairal, M. Elad, and G. Sapiro, “Sparse representation for color image restoration,” *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53–69, Jan 2008.

SYNCHRONIZATION OF INTONATION ADJUSTMENTS IN VIOLIN DUETS: TOWARDS AN OBJECTIVE EVALUATION OF MUSICAL INTERACTION

Panagiotis Papiotis, Esteban Maestre, Marco
Marchini, Alfonso Perez

Music Technology Group,
Universitat Pompeu Fabra
Barcelona, Spain

{panos.papiotis,
esteban.maestre,
marco.marchini,
alfonso.perez}@upf.edu

ABSTRACT

In ensemble music performance, such as a string quartet or duet, the musicians interact and influence each other's performance via a multitude of parameters – including tempo, dynamics, articulation of musical phrases and, depending on the type of instrument, intonation. This paper presents our ongoing research on the effect of interaction between violinists, in terms of intonation. We base our analysis on a series of experiments with professional as well as amateur musicians playing in duet and solo experimental set-ups, and then apply a series of interdependence measures on each violinist's pitch deviations from the score. Our results show that while it is possible to, solely based on intonation, distinguish between solo and duet performances for simple cases, there is a multitude of underlying factors that need to be analyzed before these techniques can be applied to more complex pieces and/or non-experimental situations.

1. INTRODUCTION

The study of music performance is a research field that is thoroughly multidisciplinary, combining elements from signal processing, computational musicology, pattern recognition, and artificial intelligence. The largest corpus of work within this field approaches the problem of performance analysis as follows: given the score for a musical piece P and a recorded performance of that piece E_p , the goal is to measure the deviations between P and E_p in terms of *timing* (onset times and note durations), *dynamics*, *articulation* and, based on the instrument type, *intonation*. Almost all existing approaches deal with the performance aspects of a single musician, and a thorough state of the art can be found in [1].

For the case of musical ensembles, such an analysis can be performed on two different levels: the *intrapersonal* level, where each musician is individually interpreting his/her own score, and the *interpersonal* level, where each musician provides and receives audiovisual feedback to/from other musicians, thus establishing a causal relationship between the performances.

This causal relationship and the influence mechanism that drives it can be perceived as an achieved *synchrony* between the musicians; taking the case of musical timing as an example, synchronized tempo curves and onset times would strongly

indicate a joint performance, as opposed to each musician performing alone and subsequently joining the recordings.

In this work, we focus on the synchronization of intonation adjustments in violins: since the violin is a fretless instrument and therefore capable of producing continuous pitch, violinists performing in a duet or larger ensemble must work together to achieve harmonic consonance for the overall sound.

1.1. Objectives

In studying the interaction mechanism that results in the synchronization of intonation adjustments, we focus on three main objectives:

- *Detecting* evidence of synchronization, as well as measuring its strength
- *Analyzing* the cause and behavior of the synchronization mechanism
- *Simulating* the mechanism by means of a computational model.

Through the achievement of these objectives, we hope to contribute in two different ways: first, by detecting and quantifying the interaction between musicians as they perform – which has potential applications both in analyzing collaborative performances as well as aiding in their realization. Second, by measuring and simulating the interaction – which can be used to synthesize audio from separate, intelligent musical agents which are aware of, and capable of adapting to each other's expressive choices in order to better control the quality of the joint acoustic result.

In this article, we investigate towards the achievement of the first objective; to detect and quantify the synchronization mechanism, specifically for the case of violin intonation. To this end, our approach was to record violinists performing their part in an interactive set-up (i.e. together) and then separately; then, using each musician's score as a reference for the expected pitch, we calculate the deviation of the performed pitch contour from the score. Finally, we attempt to measure the coupling between the pitch deviation of violinist 1 and 2, using various interdependence measures.

1.2. Related work

Regarding synchronization in musical ensembles, an important amount of the existing work has been done from a cognitive point of view, such as Keller's theory of joint action in music performance[2]. Keller focuses on three processes: *auditory imagery*, where the musician has his/her own anticipation of their own sound as well as the overall sound of the ensemble, *prioritized integrative attention*, where the musician divides his/her attention between their own actions and the actions of others, and *adaptive timing*, where the musician adjusts the performance to maintain temporal synchrony. The final process, essentially an error correction model where discrepancies between timing representations are detected, has its mathematical foundation in phase and period synchronizations; however, the main focus of this work is mainly theoretical and not immediately usable as groundwork for computational approaches.

A more practical approach on musical synchronisation can be found in [3], where the bowing gestures of two members of a skilled string quartet are studied, revealing evidence of interactive coupling in the synchronization of their movement along with a high precision in execution. Similar to Keller's approach on timing, measures of synchronization are based on approaching the musician's temporal behavior as an oscillating system, where the two musicians are coupled with each other. Another approach, similar in context but dealing more with the concept of dominance in social interaction can be found in [4].

Regarding intonation and melodic features in general, Kahlin's research[5] focused on formant frequencies in singing voice, and specifically on the effect of singing solo versus singing in a barbershop quartet, reporting that the singers strive to separate their own formants from the others; given the importance of accurate intonation in such a scenario, it is considered likely that the singers spread their formants in order to hear themselves better, which facilitates intonation.

On the specific subject of synchrony in intonation adjustments, we could not find any literature that has investigated the matter from a computational point of view. For the specific case of the violin however, there exist musicological approaches [6] and handbooks directed to violinists which discuss the hypothesis that such adjustments exist.

1.3. Outline

The remainder of this article is organized as follows: In section 2, we describe our signal acquisition process, as well as the pre-processing steps that we perform prior to our analysis; section 3 provides details about our choice for the interdependence measures; section 4 contains some experimental results of our analysis. Finally, in section 5, we discuss the results and provide some conclusions and our future directions.

2. SIGNAL ACQUISITION & PRE-PROCESSING

We conducted two series of experiments; the first round of experiments featured two professional violinists who have previous experience in performing as a duet and were familiar with the scores they were performing, while the second round of experiments featured two amateur violinists who had no experience of performing together, and without previous knowledge of the scores.

2.1. Recordings

Each piece was recorded in two discrete set-ups:

- a *solo* set-up, where each musician performed their part alone, and
- a *normal* set-up, where the musicians performed their respective part together as in a normal duet situation.

In order to reduce the complexity of the required task as well as motivate the musicians to focus on the performance unrestricted, the recordings were carried out without the use of a metronome. The pieces performed by the professional musicians were select excerpts from J.S. Bach's *Concerto for two violins* (BWV 1043) and L. Berio's *Duetti per due violini*. For the case of the amateur musicians, we opted for scores of much simpler difficulty, and thus the pieces used were the traditional piece *Greensleeves* played in unison by the two violinists, and a simplified excerpt from L. Berio's *Duetti per due violini*.

Each violinist was captured using piezoelectric pickups fitted on the bridge of the violin, while a large diaphragm condenser microphone captured the overall sound of the duet; all audio signals were captured with a sampling rate of 44100 hertz. Besides audio signals, we also recorded bowing gesture parameters using an EMF motion tracking device, using the method detailed in [7]; these signals were used to perform the audio-to-score alignment, as it is shown in the next section.

2.2. Score-performance alignment

In order to have a reference to which the intonation adjustments can be compared, it was necessary to align each performance to its respective score; this way, the score can be used as the 'expected' pitch, and the difference between this and the recorded pitch can be viewed as a score-free representation of the intonation. However, it is known that score-performance alignment is a difficult task, especially for the case of a continuous-excitation instrument such as the violin where the note onsets are varied and smooth.

Utilizing the captured bowing gestures as well as the audio information as input, we used an implementation based on the method described in [9],[10]. In this method, bow direction changes as well as more subtle measurements such as an estimation of the applied bow force provide the most probable candidates for note changes, combined with information extracted from the audio (such as the fundamental frequency curve and the root mean square energy of the recorded sound). These features are given as input to an implementation of the Viterbi algorithm, which calculates the temporal alignment between the score and the recorded performance.

2.3. Temporal alignment of experimental recordings

Since the recordings in the experimental set-ups were performed without a metronome, it was necessary to time-warp the performances in order to compare pitch deviations between violinists 1 and 2 without regarding timing information; in the solo recordings, for example, this comparison is impossible since the two recordings of violinists 1 and 2 were not temporally synchronized.

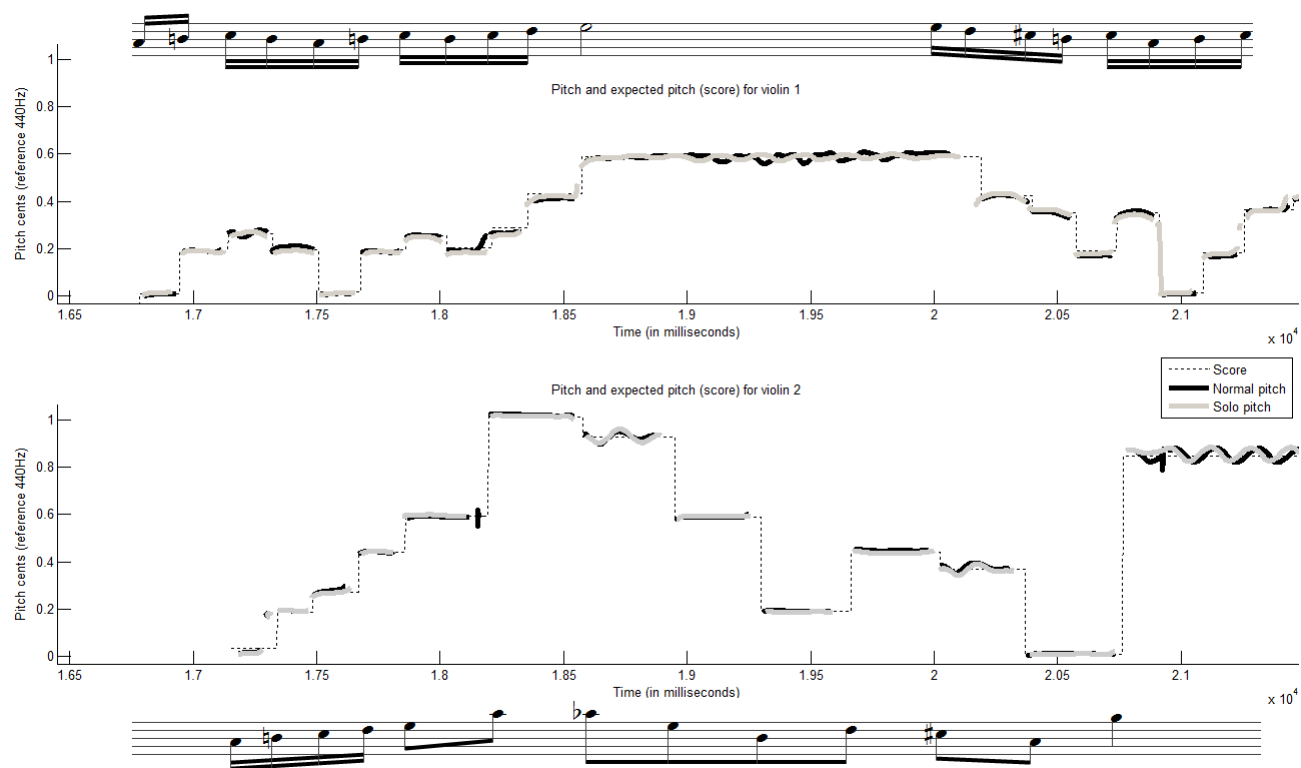


Figure 1: Expected pitch (score), recorded pitch in the normal set-up, and recorded pitch in the solo set-up, for an excerpt of J.S. Bach's 'Concerto for two violins'

This was achieved by applying a note-by-note temporal warping algorithm based on resampling the signal between note onsets and restoring its original pitch using an implementation of the phase vocoder algorithm, as described in [8]. We initially considered warping only the pitch contours instead of the recorded audio; however, besides producing an accurate and non-destructive temporal alignment (as seen in Figure 2), this approach can be also very useful in performing user evaluation tests, where subjects can hear the normal duet recordings and the solo aligned-duet recordings and rate the quality of the duet's intonation - thus investigating whether intonation adjustments alone (i.e. with no temporal mismatch) can provide enough information to discriminate between solo and duet recordings. For this reason, all recordings were warped to match the onset timings of the normal recordings, in order to preserve the natural timing of a joint performance.

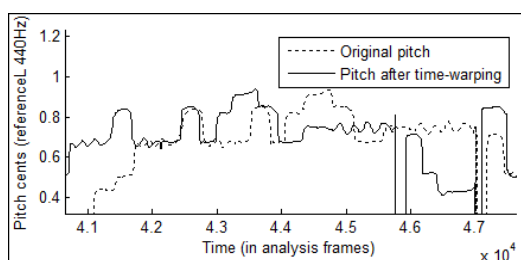


Figure 2: Extracted pitch before time-warping and after time-warping the source sound.

2.4. Audio feature extraction

The audio features that were used are the fundamental frequency contour, root mean square (RMS) energy and aperiodicity, all of which were extracted using the YIN algorithm[11]. The window size was set to 4096 samples, using a hop size of 32 samples. Both the F0 contour and the score were converted to pitch cents using 440 hertz as reference. The RMS energy and aperiodicity were used to filter out data points without a clear pitch content, while the octave errors produced from the pitch estimation were corrected using pitch guides; essentially an upper and lower bound for the F0 contour that is obtained by shifting the score-defined notes by a given threshold (± 30 pitch cents). Finally, all time series were downsampled to a sampling rate of 1KHz, in order to facilitate the comparison of time series as well as reduce the computational load for the interdependence measures used afterwards. Figure 1 presents an example of the extracted, post-processed data.

2.5. Intonation adjustments extraction

In order to extract the intonation adjustments from the F0 contours, we consider the score as a reference or 'expected pitch', i.e. perfect, non-adjusted intonation. The deviation between the recorded pitch and this reference was initially used as the intonation adjustment. However, the score is translated to fundamental frequency using the equal temperament scale, as in keyboard or fretted instruments; this does not necessarily hold

for the case of violin performance. Indeed, violinists make their choice of temperament based on a number of factors, such as the interval with the previous and next note, the string the note is being played on, as well as the type of instrument they are performing with.

This phenomenon was observed in our recordings as well; there were numerous notes in the score whose expected pitch had a systematic distance from the mean pitch performed by all violinists. Given that all measured pitch deviations varied within a very small range (within the range of 10 pitch cents), this systematically biased representation of the expected pitch introduced a lot of noise in our time series. We proceeded to bypass this problem as follows: for notes with a systematic distance larger than 5 pitch cents, we substituted the expected pitch with the mean F0 value of all performances for that particular note, in that particular piece. This choice was later validated by our results as increasing the separation between the *solo* and *normal* experimental set-ups.

3. INTERDEPENDENCE MEASURES

After extracting the pitch deviations for each violinist and each recording, the next step was to search for interdependence among the pitch deviations of violinist 1 and violinist 2. Several interdependence measures were considered and tested before we decided on which method to use. First we briefly discuss methods which did not provide useful, before finally describing the method we chose.

3.1. Preliminary interdependence measures

As a first indication to the nature of the possible interdependence between the two pitch deviations, we initially turned our attention to scatter plots between the values of the two time series. Figure 3 presents such a scatter plot, for the Greensleeves recording with the two violinists playing in *unison*, which is we considered as the simplest and clearest case of interdependence.

Linear and rank correlation. Naturally, the first obvious choice as a measure of interdependence was linear correlation. It can be observed from the scatter plots, however, that there is no visible correlation between the pitch deviations of the two violinists, and there is little difference between the solo and normal set-ups. The implications of such an observation lead us to the conclusion that, whichever synchronization phenomenon occurs is not consistent throughout the piece, and cannot be viewed as a process which is either invariant to time or cyclostationary.

Nevertheless, we calculated three correlation measures: product-moment (Pearson), Kendall and Tau correlation among the two time series. Unsurprisingly, the correlation coefficients had very low values for all experiments (≤ 0.015 average), and failed to show statistically significant separation between the *solo* and *normal* experimental set-ups.

Mutual information. Another interdependence measure that was considered was mutual information, adapted for non-bivariate time series [12]. Mutual information is a dimensionless measure first applied to Information Theory, and loosely put, measures the difference between two types of joint entropy; the joint entropy of the two variables as measured from the data, and the joint entropy of the two variables as if they were independent.

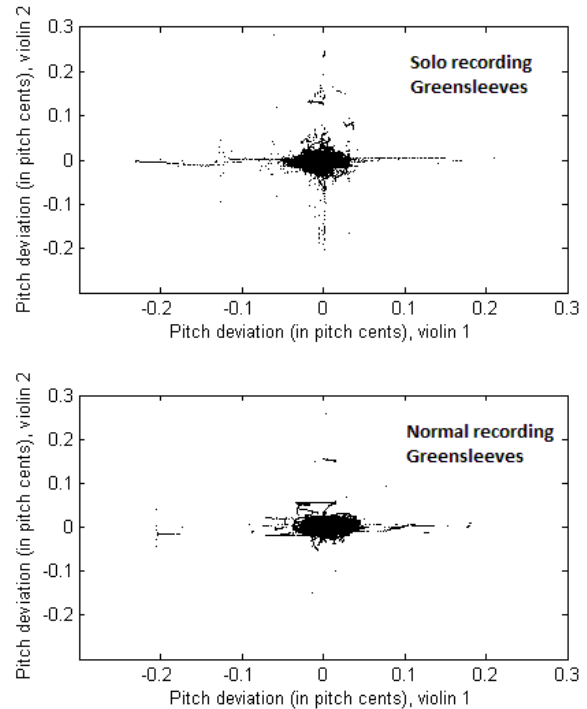


Figure 3: Scatter plots for Greensleeves between violinist 1 and 2, solo (top) and normal (bottom).

Thus, it quantifies the reduction in uncertainty about one random variable given knowledge of another. However, this measure too failed to provide separation between *normal* and *solo*, with the exception of the experiment shown in figure 3, where the violinists were playing the same melody.

Granger causality. One significant drawback of the previous measures is their lack of directionality; besides the overall degree of interdependence, it is also important to draw conclusions about the direction of influence, i.e. whether violinist 1 is influencing violinist 2 more than the opposite.

A measure that is capable of giving such an estimate is Granger causality [13], a statistical concept of causality that was first applied to Econometrics, and recently to Neuroscience. It poses the hypothesis that if variable X causes variable Y, then past values of X should significantly help in predicting future values of Y as opposed to simply using past values of Y to predict its own future, given that the data is normally distributed. The parameter that in most implementations has to be defined by the user is the maximum number of lags.

Although there are cases in auditory cognition where granger causality has been used as a measure of coupling [15], it soon became apparent that the nonlinearities of our time series were not the suitable input for this measure. The normalized causality value was very low (≤ 0.001) for all recordings and a large variety of lag values, while the separation was once again not consistent.

3.2. Non-linear coupling detection

For our final interdependence measure, we turned our attention towards methods which are widely used in computational

neuroscience. There exists a variety of nonlinear interdependence measures that quantify the signature of directional couplings among two random processes, based on distances in reconstructed state spaces. Essentially, the dynamics of each time series are reconstructed using a given number of embedding dimension and a given time delay; then, a distance matrix is calculated by estimating the distance of each point from every other in the phase space. Finally, by evaluating distances of conditioned neighbors in the distance matrix, the directional coupling for the two variables is calculated.

Of these measures, we use the measure L , which was recently shown to be of higher sensitivity and specificity for directional couplings than previous approaches. For a more in depth explanation of the method as well as its mathematical formulation, we direct the reader to [14] where the method was originally introduced.

There are four main parameters that have to be given as an input:

- the *embedding dimension* (m), which is the number of past values to be included for the state space reconstruction
- the *time delay* parameter or *tau* (τ), which is the time delay in samples between each of the past values used, and
- the number of *nearest neighbors* (k), which is the number of closest points from the distance matrix to be used for the coupling calculation, and
- the *theiler window* (W), which is the range of points to be excluded from the distance matrix in order to discard too-close neighbors

Experimenting with the values of these parameters, it became evident that the most important ones were the embedding dimension (m) and the time delay (τ), since they were the ones who had the greatest impact on the outcome of the algorithm; the number of nearest neighbors was set to 3, and the theiler window to $2 \cdot \tau$. From there on, we calculated the coupling strength between violinists 1 and 2 for each experimental recording for the following range of values for m and τ :

$$m = [2:20], \text{ and } \tau = [10:360] \text{ milliseconds, with a step of 20 ms.}$$

The rationale behind the above ranges is fairly simple: 360 milliseconds were manually identified as the maximum vibrato period in our recordings, while 20 embedding dimensions is a commonly adopted upper limit for the computation of the nonlinear coupling.

Given the number of tested values for the embedded dimension and the time delay described above, we performed 304 calculations of L for each recorded experiment; this was done to help us achieve greater statistical significance in our results.

The value of L increases with the amount of coupling strength, and is normalized between 0 and 1; higher coupling for violinist 1 than violinist 2 indicates that V1 casts a stronger influence on V2. Finally, the average coupling is calculated as the mean value of both coupling values, one for each violinist.

4. RESULTS

The results are divided in two categories – one for the recording of the amateur musicians, and one for the professional and more complex recordings.

4.1. Coupling results for amateur musicians

In all our recordings with the amateur musicians, three main empirical observations were made:

1. Violinist 2, being less adept at *prima vista*, was more focused on performing the piece correctly rather than adjusting to violinist 1. As a consequence, violinist 1 was mainly trying to adapt his intonation to violinist 2.
2. Performing a piece in unison (as in the case of *Greensleeves*) naturally exposes the mismatch in intonation the most, since the harmonic dissonance is much more apparent when the two violinists are performing the same melody, in the same tonal height; this was the case where the interaction between musicians was most evident.
3. Performing a piece where the melodic line of violin 1 is different from violin 2, made the detection of harmonic dissonance much more difficult. The same stands for the tempo of the performed piece, where slow tempos exposed bad intonation, and fast tempos also made it difficult for the musicians to keep their attention on their partner, presumably because of the cognitive load of the performance.

In Figure 4, the coupling strength for all values of m and τ is displayed as a grayscale mesh. It can be seen that, although m and τ seem to increase the overall coupling strength, the coupling value stays consistent throughout the repetitions of the L measure.

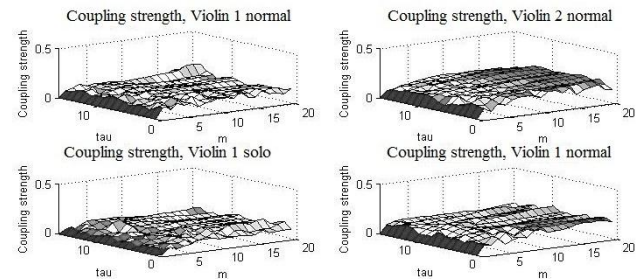


Figure 4: All calculated coupling strengths for the Berio duet recorded by the amateur violinists.

The average coupling strength for a given recording is given by taking the mean across all values of m and τ . Figure 5 shows the average coupling strength the Greensleeves recording:

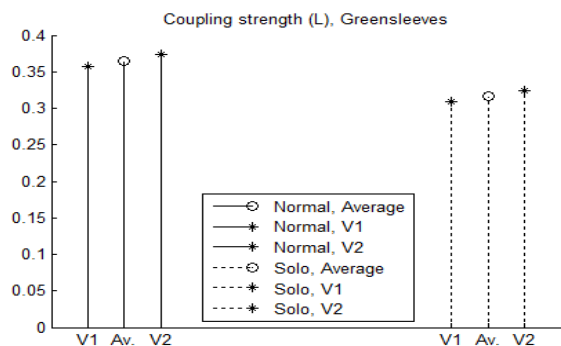


Figure 5: Overall coupling strength for *normal* and *solo* recordings of Greensleeves

Figure 6 shows the overall coupling strength for the Berio recording, featuring two repetitions of the *normal* set-up and two repetitions of the *solo* set-up:

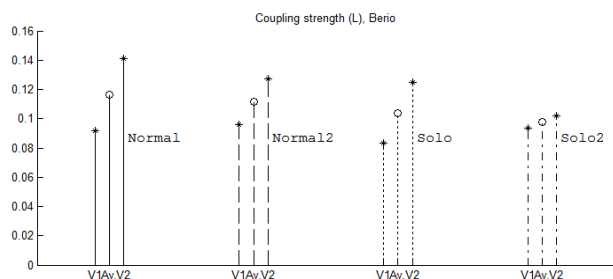


Figure 6: Overall coupling strength for two *normal* and two *solo* recordings of the L. Berio duet.

From the above calculated values, it can be seen that the coupling strength is capable of providing consistent separation between the *normal* and *solo* recordings, albeit with a small margin for more complex scores. Moreover, it is also seen that the coupling measure employed is capable of indicating the direction of the influence mechanism between the violinists; in both figures, violinist 2 has a stronger influence on violinist 1 than the opposite.

Moreover, it is observed that the coupling strength is significantly decreased when the musicians have more complex scores; in figure 5 the two violinists are performing the same score, while in figure 6 they are performing different melodic lines.

4.2. Coupling results for professional musicians

In the experiments with professional musicians, our main observation was that, since the musicians were already familiar with each other's playing, as well as the performed pieces, they could reproduce their intonation with remarkable accuracy throughout the recordings; thus shifting their attention more towards the timing and articulation aspects of the performance. This became particularly evident after listening to the time-warped recordings, where it was nearly impossible to distinguish between the *normal* recording and the *solo*. Evidence of this statement can be seen in one of the very first figures, Figure 2,

where it is evident that the *solo* and *normal* recordings have remarkably similar F0 curves.

Figure 7 shows the overall coupling strength for the Bach recording, while figure 8 shows the overall coupling strength for the L. Berio recording.

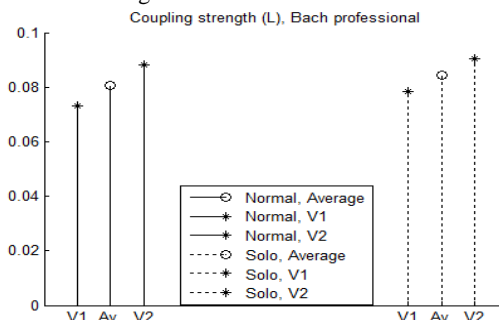


Figure 8: Overall coupling strength for *normal* and *solo* recordings of the J.S. Bach duet, with the professional musicians

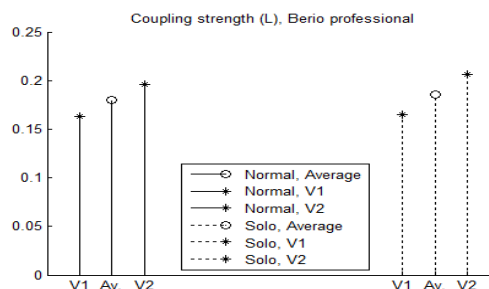


Figure 8: Overall coupling strength for *normal* and *solo* recordings of the L. Berio duet, with the professional musicians

It is evident that these cases are much harder to separate; this is attributed both to the skill of the violinists, as well as the complexity of the score. The direction of influence is consistent throughout all four recordings, although the tempo and complicated harmonic relationship between the two scores makes it difficult to validate as a result.

5. CONCLUSIONS & FUTURE WORK

In this paper we have presented a preliminary method for measuring the synchronization mechanism behind intonation adjustments in violin duets, based on the pitch deviation of each violinist from his/her respective score. An analysis procedure for the recorded material has been outlined, as well as some considerations on specific performance aspects of the violin. We tested a number of interdependence measures before concluding on the measure that provides the best results, and the final chosen measure appears to validate the hypothesis that violinists are influenced by each other's intonation when performing together, at least for the simple cases of non-professional musicians.

However, it has been seen that the coupling strength is dependent on a multitude of factors; namely the complexity of the piece, the skill of the violinists, and the harmonic relationship between the two performed melodic lines. In order to obtain clearer separation between solo and duet recordings and to study

this synchronization phenomenon without coloration from the scores, it is necessary to push towards two main improvements.

First, we believe that the use of F0 contours as the only extracted feature does not convey the real phenomenon well enough, since the pitch perception of real instruments such as the violin is strongly related to psychoacoustics factors such as the loudness and timber of the produced sound. To this end, an objective measurement of harmonic consonance/dissonance has to be included as a feature, to approach more the human perception of pitch and intonation.

Second, in order to make the coupling detection independent from the performed scores, it is necessary to post-process the scores with an algorithm that analyzes the intervals between the two violins, and adjusts the expected pitch (or the lack thereof) according to the harmonicity of the interval; this way, very harmonic intervals between the two melodic lines will greatly penalize 'bad' intonation, while inharmonic intervals will contribute much less to the coupling strength.

6. ACKNOWLEDGEMENTS

We would like to thank Dr. Ralph Andrzejak for his valuable suggestions on interdependence measures, as well as his advice and help in adapting the *L*-measure algorithm specifically for our case. We would also like to thank the anonymous reviewers for their insightful comments.

Finally, the work presented on this document has been partially supported by the EU-FP7 ICT SIEMPRE project.

7. REFERENCES

- [1] Widmer, G., & Goebel, W. : Computational Models of Expressive Music Performance: The State of the Art. *Journal of New Music Research*, 33(3), 203-216. (2004)
- [2] Keller, P. E. : Joint action in music performance. *Emerging Communication*, 10. IOS press.(2005)
- [3] Moore, G. P., & Chen, J. : Timings and interactions of skilled musicians. *Biological cybernetics*, 103(5), 401-14. (2007)
- [4] Glowinski, D., Coletta, P., Volpe, G., Camurri, A., Chiorri, C. and Schenone, A. : Multi-scale entropy analysis of dominance in social creative activities. *ACM International Conference on Multimedia*, pp. 1035-1038. (2010)
- [5] Kalin, G. : Formant Frequency Adjustment In Barbershop Quartet Singing. Doctoral dissertation, KTH, Department of Speech, Music and Hearing. (2005)
- [6] Mason, J.A. : Comparison of Solo and Ensemble Performances with Reference to Pythagorean, Just, and Equi-Tempered Intonations, *Journal of Research in Music Education* Vol. 8, No. 1 pp. 31-38 (1960)
- [7] Maestre, E., Bonada, J., Blaauw, M., Perez, A., & Gaus, E. : Acquisition of violin instrumental gestures using a commercial EMF device. *International Computer Music Conference* (p. 386–393). San Francisco (2007)
- [8] Ellis, D.P.W. : A phase vocoder in MATLAB, <http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/>, last retrieved on (March 2011)
- [9] Maestre, E. : Modeling instrumental gestures: an analysis/synthesis framework for violin bowing. Doctoral dissertation, Univ. Pompeu Fabra, Barcelona, Spain. (2009)
- [10] Perez, A : Enhancing Spectral Synthesis Techniques with Performance Gestures using the Violin as a Case Study. Doctoral dissertation, Univ. Pompeu Fabra, Barcelona, Spain. (2009)
- [11] De Cheveigné, A., & Kawahara, H. : YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*, 111(4), 1917-1930. (2002)
- [12] Dionisio, A., Menezes, R., & Mendes, D. A. : Mutual information: a measure of dependency for nonlinear time series. *Physica A: Statistical Mechanics and its Applications*, 344(1-2), 326-329. (2004)
- [13] Granger, C.W.J. : Investigating causal relations by econometric models and cross-spectral methods". *Econometrica* 37 (3), 424–438. (1969)
- [14] Chicharro D., Andrzejak R.G. : Reliable detection of directional couplings using rank statistics. *Physical Review E*, 80, 026217 (2009) S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, USA, second edition, (1991)
- [15] Londei, A., D'Ausilio, A., Basso, D., Sestieri, C., Del Gratta, C., Romani, G. L., et al. Brain network for passive word listening as evaluated with ICA and Granger causality. *Brain Research Bulletin*, 72(4-6), 284-292. (2007)

GMM SUPERVECTOR FOR CONTENT BASED MUSIC SIMILARITY

*Christophe Charbuillet, Damien Tardieu and Geoffroy Peeters, **

IRCAM-STMS-CNRS

Centre Pompidou

Paris, France

charbuillet, tardieu, peeters (at) ircam.fr

ABSTRACT

Timbral modeling is fundamental in content based music similarity systems. It is usually achieved by modeling the short term features by a Gaussian Model (GM) or Gaussian Mixture Models (GMM). In this article we propose to achieve this goal by using the GMM-supervector approach. This method allows to represent complex statistical models by an Euclidean vector. Experiments performed for the music similarity task showed that this model outperform state of the art approaches. Moreover, it reduces the similarity search time by a factor of ≈ 100 compared to state of the art GM modeling. Furthermore, we propose a new supervector normalization which makes the GMM-supervector approach more preformant for the music similarity task. The proposed normalization can be applied to other Euclidean models.

1. INTRODUCTION

Exploring the wide world of music requires some navigation tools. To discover new tracks, one might consider several options. Specialized magazines or music expert friends can guide the user. In a more passive way, the user can wait for new music production by listening to his favorite radio or following the statistically made recommendation of online mp3 providers, based on user profiles and purchase analyses. But to explore several million of iTunes[®] music tracks, one may need to employ a content based similarity search system. The principle is quite simple. From a starting music and for a given similarity measure, the system provides the user a list of similar songs found in the entire database. If the user is not satisfied with the result, he/she can change or adapt the similarity measure according to his/her wishes. The system can also learn the user preferences using relevance feedback. One can also use the result of previous queries as starting point for a new search and, thereby, perform a step by step smart exploration of the music space.

Obviously, the relevance of the similarity measure is fundamental. A music track can be described in several ways. Using the mpeg-7 taxonomy, we distinguish the meta description (e.g.: music author or title) and the content description. Similarity systems based on content description mimic human perception of similarity. Timbral modeling is nowadays state of the art in such systems. It consists in statistical modeling of short term audio features, usually the Mel Frequency Cepstrum Coefficients (MFCC). The model used can be a Gaussian Mixture Model (GMM) as proposed in [1, 2], or a single Gaussian Model with full covariance matrix [3, 4] which provides similar performances. The measure

used to compare the models is the Symmetrized Kullback-Leibler Divergence (SKLD) [5] or alternatively the Earth Mover's Distance based on the SKLD when models are GMMs [2].

We present here an application of the Gaussian Mixture Model using Universal Background Model (GMM-UBM) approach for content based music similarity. This method, initially developed in the field of speaker recognition [6] has been successfully applied for music genre classification and similarity [7]. The main idea is to build a generic Gaussian mixture model by using a large data set of representative signals, which are in our case extracted from a large set of music tracks. This model, named Universal Background Model, aims at modeling the overall data distribution and can be composed of hundred of Gaussians. The model for a specific track is then obtained by adapting the UBM model parameters by using the track data. The final model is composed of a subset of the GMM parameters, stacked into a vector, the so called supervector. This approach presents several advantages:

- it allows to build a complex model from a small amount of data,
- the final model can be embedded into the Euclidean space, which allows fast similarity search.

In this paper, a complete description of the GMM-UBM model is proposed in section 2. Then our main contribution is presented in section 3. It consists in a new supervector transformation, which provides a significant improvement of the similarity system. Experiments are detailed in section 4 and the perspectives of this work are presented in section 5.

2. GMM-UBM APPROACH

2.1. Universal Background Model

The Universal Background Model (UBM) aims at modeling the overall data distribution. It consists of a classical Gaussian Mixture Model. For a D-dimensional feature vector \mathbf{x} the mixture density used for the likelihood function is defined as a weighted sum of unimodal Gaussian models :

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^M \omega_i p_i(\mathbf{x}) \quad (1)$$

where M is the number of Gaussian components, $p_i = \mathcal{N}(\mu_i, \Sigma_i)$. λ represents the GMM parameters, where $\lambda_i = \{\omega_i, \mu_i, \Sigma_i\}$, $i = 1, \dots, M$. \mathbf{x} represents a feature vector, which in our case is a short term descriptor, usually an MFCC.

* This work was supported by the French Oseo project QUAERO

The UBM is usually composed of Gaussian models with diagonal covariance matrix. The loss of modeling ability due the diagonal covariance matrix can be compensated by increasing the number of Gaussian in the mixture [6]. The UBM is trained using a large and representative set of data by using the Expectation Maximization (EM) algorithm.

2.2. UBM adaptation

The UBM adaptation is the process of modifying the UBM parameters in order to fit a particular data distribution. In our application, this subset is the data extracted from a track to modelize.

This adaptation is made using the Maximum A Posteriori (MAP) approach. The first step consists in determining the probabilistic alignment of the training vectors with the UBM Gaussian components. For a Gaussian i in the UBM we compute :

$$\mathbf{Pr}(i, \mathbf{x}_t) = \frac{\omega_i p_i(\mathbf{x}_t)}{\sum_{j=1}^M \omega_j p_j(\mathbf{x}_t)} \quad (2)$$

$$n_i = \sum_{t=1}^T \mathbf{Pr}(i, \mathbf{x}_t) \quad (3)$$

$$E_i(\mathbf{x}) = \frac{1}{n_i} \sum_{t=1}^T \mathbf{Pr}(i, \mathbf{x}_t) \mathbf{x}_t \quad (4)$$

These statistical values are then used for adapting the mean vector $\hat{\mu}$ of each Gaussian during the following iterative process:

$$\hat{\mu}_i^0 = \mu_i \quad (5)$$

$$\hat{\mu}_i^k = \alpha_i E_i(\mathbf{x}) + (1 - \alpha_i) \hat{\mu}_i^{k-1} \quad (6)$$

$$\alpha_i = \frac{n_i}{n_i + r} \quad (7)$$

where \mathbf{x}_t represents the t^{th} feature vector of the music track to modelize and r is a fixed "relevance factor", usually set between 8 and 20. $k = 1, \dots, K$ represents the iteration number.

2.3. GMM supervector

To summarize, a music track model is directly derived from a generic GMM, estimated using a large set of representative data (the so called UBM). During the adaptation process, only the mean vectors of the Gaussians are modified to fit the particular music track distribution. Consequently, all the the music track models have both the same covariance matrix and weight. Knowing the parameter of the UBM, a particular music model can be summarized by the mean vectors of its Gaussian mixture components:

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_N \end{pmatrix} \quad (8)$$

where μ , named the GMM supervector, is the concatenation of all the mean vectors of the N Gaussian components. In [8], the authors propose to approximate the Kullback-Leibler divergence between two models a and b by :

$$d(\mu^a, \mu^b) = \frac{1}{2} \sum_{i=1}^N \omega_i (\mu_i^a - \mu_i^b)^T \Sigma_i^{-1} (\mu_i^a - \mu_i^b) \quad (9)$$

where μ^a and μ^b are the GMM supervectors of the models a and b respectively, λ_i represents the mixture weights and Σ_i the covariance matrix of the i^{th} Gaussian component (which is common to the models a and b). From this representation, we can deduce the following natural normalization:

$$\bar{\mu}_i = \sqrt{\omega_i} \Sigma_i^{-1/2} \mu_i \quad (10)$$

$$i \in 1, \dots, N$$

where N is the number of Gaussian components of the model. Then, the divergence presented in eq. 9 can be rewritten as the square Euclidean distance between the normalized supervectors:

$$d(\mu^a, \mu^b) = \frac{1}{2} \|\bar{\mu}^a - \bar{\mu}^b\|^2 \quad (11)$$

Finally, because of the monotony of the square function $(\cdot)^2$, one can directly use the Euclidean distance $\|\bar{\mu}^a - \bar{\mu}^b\|$ for music similarity retrieval, as proposed in [7].

3. SUPERVECTOR NORMALIZATION FOR MUSIC SIMILARITY

3.1. Hubs and orphans

Even if the statistical modeling of short term descriptors gives good results for music similarity, it usually tends to create false positive results which are usually the same songs. This songs, named hubs, are falsely close to all the tracks of the database. As well, some songs, named orphans, are falsely far from the rest of the database. J. Aucouturier *et al.* [9] showed that this phenomenon is "not a property of a given modeling strategy and tends to appear with any type of model".

For a better understanding of the problem we propose to modelize a set of music tracks and to study their distance distributions. The music database and the modeling process are fully described in section 4. From this set of supervectors we compute the distance matrix between all the supervectors. Figure 1 presents the distance distribution between the track supervectors and the rest of the database. We can observe that the distributions have a significant variability. For example, the distribution related to the first music track shows that this model is far from the rest of the database. Consequently, it will have a poor probability to appear within the results of the similarity search. This is a good example of an "orphan" song.

3.2. P-norm

To overcome this drawback, a distance normalization method was proposed by T. Pohle *et al.* in [4]. The key idea of this method is to transform the distances between two models by using their distance distribution according to a normalization set. After the normalization process, the histogram of the new "distance" between

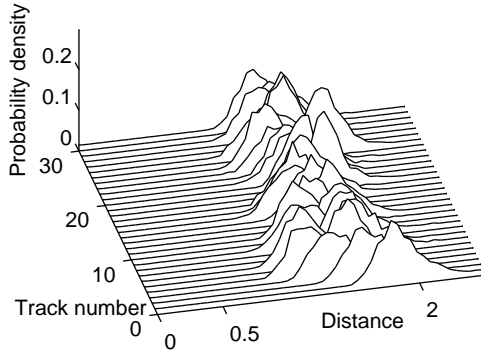


Figure 1: Distance distribution between supervectors. Each curve represents the histogram of the distance between a given supervector and the rest of the database.

a model and the rest of the database must be a normal distribution $\mathcal{N}(0, 1)$. This normalization is given by:

$$\text{P-norm}(d(a, b)) = \frac{1}{2} \left(\frac{d(a, b) - \hat{\mu}_a}{\hat{\sigma}_a} + \frac{d(a, b) - \hat{\mu}_b}{\hat{\sigma}_b} \right) \quad (12)$$

where $d(a, b)$ is the original distance between models a and b , $\hat{\mu}_a, \hat{\sigma}_a$ are the mean and standard deviation of the distances between the models a and the normalization set. For convenience, in the rest of this paper, we will refer to this method as the P-norm (Pohle-normalization). One can notice that this type of normalization is very close to the ZT-norm developed in the field of speaker verification [10].

3.3. UCS and MCS normalizations

An important benefit of the supervector approach is the ability to represent a complex statistical model as an Euclidean vector. It allows the use of efficient indexing algorithms for fast similarity search into very large databases like local sensitive hashing [11]. The use of the P-norm (which modifies the distances) transforms the original Euclidean space into a non-metric space, constraining the use of ad-hoc indexing methods which are usually slower. Therefore, a normalization which can be applied directly to the supervector is more suitable. Let us consider the supervector as a point into a high dimensional space and a large representative data set. To reproduce the benefit of the P-norm by a geometric transformation of the supervectors, the projected points must “see the world in a same way” i.e. the distance distribution between a point and the rest of the database must be the same for all the points. It is easy to show that a uniform data distribution on a hyper sphere satisfies this constraint. We propose two different methods to reach this goal:

1. project the supervectors on a unit sphere centered on the Universal Background Model,
2. project the supervectors on a unit sphere centered on the mean supervector of a representative data set (here we used the entire database).

For convenience, we named the first approach the *UBM Centered Spherical normalization* (UCS-norm) and the second one the *Mean*

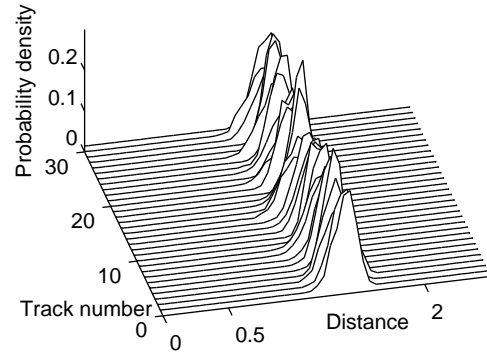


Figure 2: Distance distribution between supervectors normalized by the UCS-norm.

Centered Spherical normalization (MCS-norm). The following equations detail the implementation we used.

$$\bar{\mu}_{UCS}^a = \frac{\bar{\mu}^a - \bar{\mu}^{UBM}}{\|\bar{\mu}^a - \bar{\mu}^{UBM}\|} \quad (13)$$

$$\bar{\mu}_{MCS}^a = \frac{\bar{\mu}^a - \mathcal{M}}{\|\bar{\mu}^a - \mathcal{M}\|} \quad (14)$$

$$\text{with } \mathcal{M} = \frac{1}{N} \sum_{k=1}^N \bar{\mu}^k, \bar{\mu}^k \in \Omega \quad (15)$$

where $\bar{\mu}^a$ represents the supervector to normalize, $\bar{\mu}^{UBM}$ is the supervector of the UBM and \mathcal{M} represents the mean supervector of a subset Ω composed of N supervectors. Figure 2 clearly shows that the UCS-norm allows to reduce the variability of the distance distributions. One can observe that the track number one is no more “orphan” still its distances from the rest of the database have been significantly reduced.

4. EXPERIMENTS

4.1. Data set

For our experiments, we used a music data set composed of 1304 tracks belonging to the following music genres: Country, Electronica, Folk, Gospel, Jazz, Latin, New Age, Pop/Rock, R&B, Rap, Reggae, World. These songs, originally encoded in mp3 32kHz stereo were down-sampled in 22050 kHz and turned into mono by summing the two channels.

4.2. Feature extraction

The short term feature vectors extracted are composed of 13 Mel Frequency Cepstrum Coefficients (MFCC) and 4 Spectral Flatness Measures (SFM). This extraction is made using a sliding window of 40 ms and a hop size of 20 ms.

4.3. Model

For this experiment, we used two types of models: the GMM supervector and a classical multivariate Gaussian Model (GM) with full covariance matrix. For the GMM-UBM approach, the whole data set was used for building a UBM composed of 64 Gaussian components with diagonal covariance matrix. This model was adapted for each song with 5 iterations of MAP using a relevance factor $r = 10$ (see 2.2). Normalized supervectors were extracted as described in section 2.3. The similarity is obtained by the Euclidean distance between supervectors. In the case of GM, the Symmetrized Kullback-Leibler divergence is used.

4.4. Evaluation metric

The evaluation metric used is the “average ratio of genre matches” in the top 1, 3 and 5 nearest neighbors after filtering the results belonging to the same artist as proposed in the MIREX Audio Music Similarity and Retrieval task¹.

4.5. Similarity search time cost

For the GM approach we used a fast implementation of the Symetrized Kullback-Leibler Divergence using its close form expression for multivariate Gaussain models. The covariance matrix inversion was computed off-line and stored into the model. With this system, the time cost for computing the full similarity matrix was of 16 s in a 3GHz 64bits computer which represents $\approx 9.4 \cdot 10^{-6}$ s by model comparison. Using the supervector approach, the duration of the entire similarity matrix process was of 0.13 s, which represents $\approx 7.6 \cdot 10^{-8}$ s by model comparison, representing a time improvement factor of 123.

4.6. Results

The obtained similarity results are presented in Table 1. First of all, we can observe that the supervector approach is slightly better than the standard Gaussian Model using the Kullback-Leibler divergence when no normalization is used. We can also notice the relevance of the P-norm. The UCS-norm and MCS-norm when applied for supervector normalization allows a significant performance improvement compare to the supervector whithout normalization. Moreover, the proposed normalizations methods perform slightly better in average than the P-norm. It is interesting to notice that the MCS-norm achives a better normalization than the UBM centered one. Furthermore, chaining the UCS-norm and the MCS-norm (SV + UCS-norm + MCS-norm) and using a sequence of all the normalization methods (SV + UCS-norm + MCS-norm + P-norm) significantly improve the results, showing that these normalization methods are complementary.

5. CONCLUSIONS

We have presented here an application of GMM supervector approach to the music similarity task. This modeling method allows to represent a complex statistical distribution into a Euclidean vector. We have proposed two new supervector projections suitable for the music similarity task. Experiments showed the relevance of our approach.

¹<http://www.music-ir.org/mirex>

Table 1: Average ratio of artist-filtered genre matches in the top 1, 3 and 5 nearest neighbors. GM = Gaussian Model, SV = Supervector. The last column shows the type of distance related.

System	1NN	3NN	5NN	dist. type
GM	45.01	44.06	44.20	non eucl.
GM + P-norm	48.31	47.52	47.14	non eucl.
SV	46.93	45.67	45.07	euclidean
SV + P-norm	51.38	49.16	47.95	non eucl.
SV + UCS-norm	50.15	49.13	48.81	euclidean
SV + MCS-norm	50.92	49.80	49.09	euclidean
SV + UCS-norm + MCS-norm	51.08	50.13	49.45	euclidean
SV + UCS-norm + MCS-norm + P-norm	52.61	51.51	50.41	non eucl.

The improvement obtained by the MCS-norm is promising. Indeed, this normalization can be applied to all type of models which can be embedded into the Euclidean space. Our current research focuses on extending this normalization to other type of models.

6. REFERENCES

- [1] J J Aucouturier and F Pachet, “Finding songs that sound the same,” in *Proc. of IEEE Benelux Workshop on Model based Processing and Coding of Audio*, 2002, pp. 1–8.
- [2] B. Logan and a. Salomon, “A music similarity function based on signal analysis,” in *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001*, 2001, vol. 00, pp. 745–748, IEEE.
- [3] M. Mandel and D. Ellis, “Song-level features and support vector machines for music classification,” in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.
- [4] T. Pohle and D. Schnitzer, “Striving for an improved audio similarity measure,” in *4th Annual Music Information Retrieval Evaluation Exchange*, 2007.
- [5] S. Kullback, *Information theory and statistics*, Wiley Publication in Mathematical Statistics, 1959.
- [6] DA Reynolds, TF Quatieri, and RB Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital signal processing*, 2000.
- [7] C. Cao and M. Li, “Thinkits submissions for MIREX 2009 audio music classification and similarity tasks,” in *MIREX abstracts, International Conference on Music Information Retrieval*, 2009.
- [8] W M Campbell, D E Sturim, and D A Reynolds, “Support Vector Machines Using GMM Supervectors for Speaker Verification,” *IEEE SIGNAL PROCESSING LETTERS*, vol. 13, no. 5, pp. 308, 2006.
- [9] J.J. Aucouturier and F. Pachet, “A scale-free distribution of false positives for a large class of audio similarity measures,” *Pattern Recognition*, vol. 41, no. 1, pp. 272–284, 2008.
- [10] R Auckenthaler, M Carey, and H Lloyd-Thomas, “Score normalization for text-independent speaker verification systems,” *Digital Signal Processing*, 2000.
- [11] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” in *SCG'04: Proceedings of the twentieth annual symposium on Computational geometry*, New York, NY, USA, 2004, pp. 253–262, ACM.

AUTOMATIC ALIGNMENT OF AUDIO OCCURRENCES: APPLICATION TO THE VERIFICATION AND SYNCHRONIZATION OF AUDIO FINGERPRINTING ANNOTATION

Mathieu Ramona

IRCAM, Centre Pompidou
1, place Igor Stravinsky, 75004 Paris, France
mathieu.ramona@ircam.fr

Geoffroy Peeters

IRCAM, Centre Pompidou
1, place Igor Stravinsky, 75004 Paris, France
geoffroy.peeters@ircam.fr

ABSTRACT

We propose here an original method for the automatic alignment of temporally distorted occurrences of audio items. The method is based on a so-called *item-restricted fingerprinting* process and a segment detection scheme. The high-precision estimation of the temporal distortions allows to compensate these alterations and obtain a perfect synchronization between the original item and the altered occurrence. Among the applications of this process, we focus on the verification and the alignment of audio fingerprinting annotations. Perceptual evaluation confirms the efficiency of the method in detecting wrong annotations, and confirms the high precision of the synchronization on the occurrences.

1. INTRODUCTION

Audio identification aims at detecting occurrences of known audio tracks in an unknown audio signal or stream. A typical example is the identification of songs in a broadcast recording. An "occurrence" is defined as the presence in an unknown signal of a degraded, or modified, version of an original audio track, that remains recognizable. An audio identification system typically searches for the possible occurrences of a large collection of tracks previously "learned" in a database. Each track contained in the database is called an audio "item".

One of the main challenges of audio identification is the robustness to the possible degradations between the original item signal and the occurrence signal. Typical degradations are audio encodings (MPEG, Real Audio...), filterings, noise addition, etc. These degradations do not alter the temporal evolution of the signal. On the opposite, the so-called *dynamic* degradations, such as time-scale changes, or signal cropping, induce a loss of alignment between the original item and the occurrence. The problem answered by this paper is to estimate very precisely these dynamic degradations, in order to realign both signals. The motivations for this issue will be explained later on.

The subject of audio alignment has been covered in the past, especially in the domaine of audio-to-score alignment, which consists in aligning the audio signal of an execution of a musical piece with the score itself. Joder et al. [1] and Cont [2] both answer this problem with Hidden Markov Models coupled with a tempo model. Müller et al. [3] also propose an algorithm to align a performance of a song on the score, based on a chroma representation. However, these approaches all deal with the alignment with respect to a score, i.e. a symbolic representation of music, whereas the problem here is the alignment of two audio signals. Müller et al. [4] also applied the chroma representation for the matching of two musical interpretations (i.e. two audio signals) of the same

piece. But their contribution only focuses on the detection of these matches, not on their temporal alignment.

The method we propose here for the alignment of audio occurrences is based on an original scheme, derived from the audio fingerprinting technique. Indeed, in [5], Casey et al. rank the problems of *audio queries* by order of similarity between the query and the reference. While genre classification and cover song detection connect very different audio signals from their *semantic* musical content, audio fingerprinting is described as "identifying a recording in the presence of a distorting communicating channel". Audio fingerprinting is in fact one of the main methods (along with audio watermarking) to perform audio identification. It consists in computing perceptually relevant numerical codes (the so-called *fingerprints*) that characterize the signal of the audio items of the database. When performing identification, similar codes are computed from the unknown signal, are compared to the codes stored in the database. This similarity search allows to identify the occurrences of the items in the unknown signal.

This paper will show how an audio fingerprinting technology can be exploited for the automatic alignment of audio occurrences, and consequently, for the correction and the refinement of ground truth annotations for audio identification evaluation. We will explain thoroughly in Section 2 why there is a need for such an automated process of annotation verification in the context of audio fingerprinting evaluation. We then present in detail the context and the terminology of the problem in Section 3, before presenting the alignment process in Section 4. A first application of this process on the Quaero audio identification corpus is presented and commented in Section 5, along with some audio examples. Then we comment on the applications and perspectives of this contribution in Section 6.

2. AUDIO FINGERPRINTING EVALUATION

Research on audio fingerprinting has been very active in the last ten years, and commercial applications based on this technology are numerous. However, contrary to other subjects in audio indexing, there is no consensus on the evaluation protocol, nor any public evaluation campaign for audio fingerprinting. One might argue that the main reason that the main commercial system already work very well. However, most companies actually have not published results of their systems on a large public database. The Quaero project has brought a first step in this direction with its first evaluation campaign for audio identification that was held in September 2010. Following this campaign, a collaborative paper from the participants was submitted [6], that discusses the issues of audio fingerprinting evaluation and proposes a public evaluation framework (available at <http://pyafe.niderb.fr/>).

Related works on evaluation The main obstacle in audio fingerprinting evaluation lies in the cost of collecting a large real-world corpus, with reliable and precise annotations. The Quaero evaluation campaign is based on a musical track monitoring scenario and is based on a corpus, provided by Yacast¹, containing real-world radio broadcast audio data. Cano et al. [7] also evaluate the identification rate of musical tracks on 12h broadcasted audio streams, but most of the authors measure the robustness of the fingerprinting code on audio items altered by typical audio degradations. Wang (Shazam) [8] evaluates the recognition rate over 250 items, after a GSM compression step and under the addition of noise with controlled SNR. Additive noise over clean items is also used by Weinstein and Moreno (Google) [9]. Haitsma and Kalker (Philips) [10] also propose a protocol involving a large collection of audio degradations (MP3 or Real Media encoding, equalization, various filters, time-scale modification, etc.) applied to four clean items.

Artificial vs Real-world distortions Artificial alterations are indeed preferred in the literature for several reasons: the corpus only consists in a collection of audio tracks and is thus much more easy to collect, the alterations are easily applied, and, most of all, they can be precisely controlled. It is then possible to study the evolution of the robustness with regard to the SNR, or the time-scaling factor. On the other hand, these artificial degradations do not reflect real-world situations. Indeed, in a typical case of broadcast emission, any musical track is generally slightly time-scaled, dynamically compressed, affected by additional noise, and subject to MP3-like encoding or digital-analog conversion.

Real-world annotation Nevertheless, a corpus based on real-world data needs the human annotation of all the occurrences of the items in the stream, and most of the degradation process is unknown to the experimenter. Annotating the start and end times of an occurrence is easy, but the annotation of a large scale corpus will generally imply a low precision (even a one second precision is ambitious). Moreover, it is almost impossible to determine manually the time-scale factor. Finally, a certain amount of mistakes is expected from manual annotation, especially when the item collection involves different edits of the same song.

The method proposed here is an ideal mean for verifying and improving such manual annotations, as we will show in this paper. The detection of missing occurrences is not in the scope of this article, but the detection of wrongly annotated occurrences proves very efficient. The alignment of the occurrence signal with the original item signal (and thus of the fingerprint codes computed from both signals) allows the application of several evaluation schemes used on artificial corpuses. We hope that this new type of annotation post-processing will encourage evaluations of audio fingerprinting techniques on real-world corpuses.

3. CONTEXT

The problem that is raised here is similar to that of audio identification, as explained before: occurrences of known audio items are to be found and located in an audio stream. However, we seek

here a much more precise result, which is made possible by the use of prior information, unavailable in a common audio identification scenario. We suppose here that the processed signal has been previously annotated, either manually or during the production of an artificial corpus. In both cases the annotation may be unprecise, and only consists of a collection of item occurrences in the audio streams, characterized by the item index in the database, and the approximate start and end times in the stream. The annotated times can be wrong by a few seconds, the error being compensated by a larger analysis scope.

An audio item can even be a sample of a song (for instance the chorus), instead of the whole track, and its exact position in the song unknown. This situation remains equivalent to using the whole song, as long as the scope of analysis includes the whole song. However, this implies, since a musical track structure is generally repetitive, that the excerpt, or a part of it, may be detected several times in the scope of analysis. Such repetitions must be discarded from the alignment process, in order to focus on the most reliable occurrence.

The item occurrence in the stream can be affected by typical audio distortions, either artificially generated or sampled from a real-world corpus. These distortions fall into two categories:

Static distortions: do not affect the temporality of the signal, i.e. the original and distorted signals are perceived as synchronous on their whole scope. Typical examples are linear filters, equalization, amplification, analog/digital conversions, typical audio encodings (MPEG, OGG ...), noise addition, loudspeaker/microphone loop.

Temporal distortions: affect the synchronization between the original and distorted signals. The process proposed here intends to estimate precisely these temporal distortions in order to be able to correct them and reach a perfect alignment between the original and distorted signals. Temporal distortions considered here are:

- *Shifting*: in the case of frame-sequence analysis, a slight shift (of a few tenth of seconds) between the signals can induce major differences in the content of the frames. It is thus important to synchronize the start time of both signals.
- *Scaling*: for instance, radio stations very often accelerate or slow down musical tracks to fit in a live schedule.
- *Cropping*: the beginning or the end of the audio item can be absent from the occurrence.
- *Insertions*: although this distortion is less common, the item can be interrupted by another signal, and then played again from where it stopped. This induces a slight shift between the item and the occurrence, that also requires a proper correction (i.e. cutting the inserted signal).

Figure 1 sums up the temporal characteristics that we intend to evaluate in this process:

1. **ItemTime**: the time in the stream that corresponds to the beginning of the database item.
2. **StartTime**: the time, relative to the ItemTime, where the occurrence actually starts in the stream. It is positive or zero. When strictly positive, it means that a part of the item beginning is not played in the stream.
3. **EndTime**: the time, relative to the ItemTime, where the occurrence ends in the stream. It is strictly positive, and upper bounded by the ItemDuration \times TimeFactor, when the item is played until its end.

¹<http://www.yacast.fr/fr/index.html>

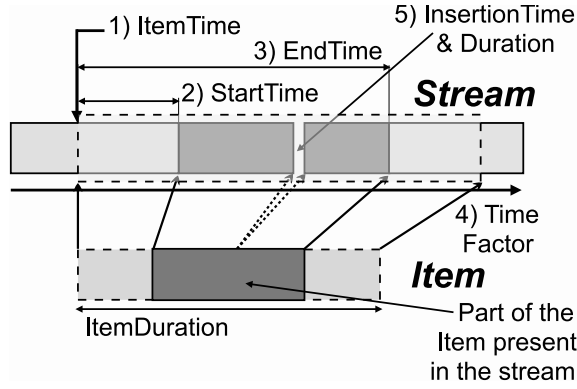


Figure 1: Characterization of the temporal alterations encountered on the occurrence (at the top) of a given item (at the bottom).

4. **TimeFactor:** ratio between a time relative to the ItemTime in the stream, and the corresponding time in the item signal.
5. **InsertTime & InsertDuration:** if the insertion happens in the stream (as in Figure 1), the time is relative to the ItemTime ; if it happens in the item, the time is absolute in the item. The same holds for the duration. Several insertions can be observed on a single occurrence.

The ItemDuration (also denoted D) is the duration of the full item signal. All characteristics are expressed in seconds.

4. DESCRIPTION OF THE ALIGNMENT PROCESS

4.1. Item-restricted fingerprinting

The key element of the alignment process is the application of a customized *Item-restricted* fingerprinting technique that we present here.

The normal process of audio fingerprinting is to fill a database with the fingerprint codes computed from a large collection of audio items. Each item is described by multiple code, computed at various time position in the signal, in order to be able to recognize any subset of it. In the present context, for each occurrence, the item is already known (from the annotation). So for each occurrence, a new database is built specifically, that contains only the fingerprint codes computed from this item. This constraint dramatically reduces the size of the database, and thus the search time. Consequently, the result of the search is a sequence of timestamps describing the position of the codes in the original item.

The use of a fingerprinting method ensures a sufficient robustness to static distortions observed in the stream occurrence. The fingerprint method used here is the one developed by the Ircam [11]. It is based on a double-nested Short Term Fourier Transform of the audio signal, over overlapping frames of a few seconds. The original method has recently been upgraded [12] with perceptual scales (a Bark filter-bank for the short-term FFT and a sone scale for the amplitudes of the long-term FFT). The resulting code is a real vector of 36 components.

The latter article [12] also describes an upgrade of the algorithm based on onset detection, but this part of the algorithm is not used here, and we rely on a regular frame scheme. In order to locally reduce the effect of the temporal distortions, the frame

size is kept relatively short (2 s). The hop size is much shorter (50 ms), than in the "standard" fingerprint process (originally set to 0.5 s). This implies that the expected temporal shift between corresponding codes is 12.5 m, which represents a negligible portion (0.6%) of the frame size. Theoretically, any other fingerprinting method could be adapted to this item-restricted scheme, but the Ircam is preferred, precisely because it involves larger window and step sizes that most methods, and thus reduces the number of fingerprint codes per item.

The first step of the algorithm consists in computing the fingerprint codes for each item, and storing them with the corresponding timestamps. Each minute of signal generates about 1200 codes. For each annotated occurrence in the stream, a sequence of codes is computed on the scope of analysis. Then a simple nearest neighbor search ($k = 1$ neighbor) is performed among the codes of the item, to collect the resulting sequence of timestamps associated. The so-called *timestamp sequence* is denoted by a set $(x_i, y_i)_{i=1, \dots, n}$, where n is the number of frames, and x_i and y_i are respectively the time of the frame in the stream and the timestamp of the nearest neighbor in the item.

Figure 2 shows several examples of timestamp sequences. The ideal detection of the full item, illustrated by Figure 2(a), implies the presence of a solid line segment of slope close to 1, that binds the ordinates 0 and D (in our case all the audio items are 60 s long). Another fragmented line is also visible on the figure, that denotes a repetition of the item, with alterations. On the opposite, Figure 2(b) shows a clear example of wrong annotation, where the dots are randomly drawn. The dot distribution is clearly not uniform though, and shows higher densities on some constant ordinate lines. This is a simple expression of the classical phenomenon in similarity of "hubs" [13], i.e. examples that are near to all other examples in a distribution. Figure 2(c) shows an example of cropped occurrence where the beginning of the item is missing. Figure 2(d) shows a line degraded in the beginning, illustrating an example of occurrence where the first seconds are covered by another signal (possibly the radio host voice). Figure 2(e) shows a clear example of insertion of a signal snippet in the middle of the original item signal. Finally, Figure 2(f) is an interesting example of fragmentation of the line, that denotes the detection of separated chunks of the original item, probably because of an edit mismatch between the item and the stream occurrence. Indeed, radio stations frequently use specific edits of a song with structures that greatly differ from the original album edit.

These examples show that the alignment process basically consists in a segment detection algorithm in the timestamp sequence. Each segment is described by the following equation:

$$y = ax + b \quad \forall x \in [x_{\text{start}}; x_{\text{end}}], \quad (1)$$

where

- a is the slope of the line containing the segment, that correspond to the time-factor previously introduced.
- b is the offset of the line containing the segment. Differences between the b values will indicate the presence of insertions.
- $[x_{\text{start}}; x_{\text{end}}]$ are the boundaries in abscissa of the segment. These will determine the ItemTime, StartTime, CutTime and EndTime values introduced in Figure 1.

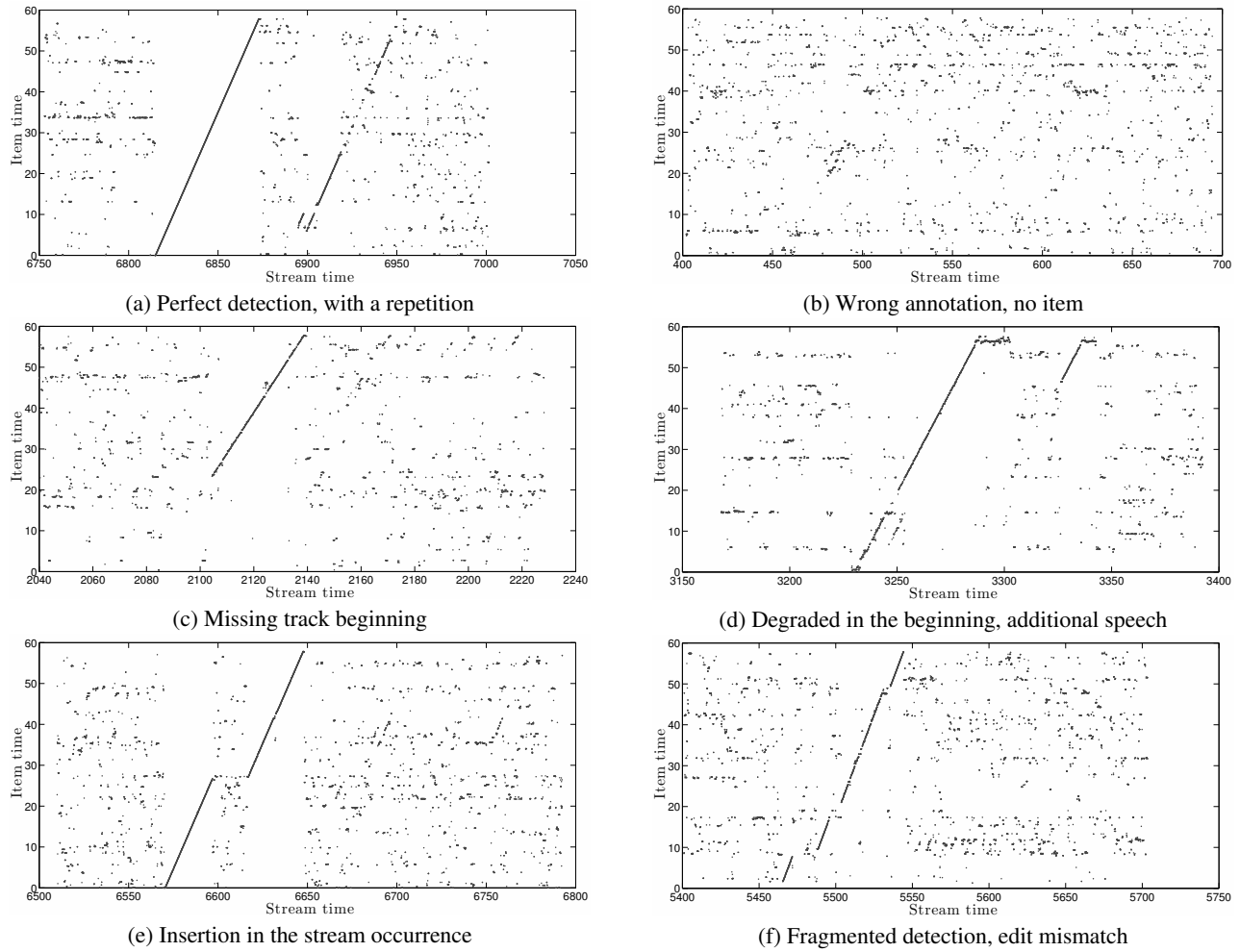


Figure 2: *Timestamp sequences: ordinates represent the time in the item of the nearest neighbor to the code computed from the stream at the time in abscissa. Several examples of result are shown, ranging from the ideal detection of the item (a) to the absence of detection (b).*

A Hough transform [14] could of course be used for this line detection problem. However, it is more costly than the method proposed below. Moreover, as we will show, the evaluation of the common slope is done jointly on all the segments and is therefore more robust and precise than a fusion of separate estimations from each segment.

4.2. Time-factor estimation

The first step consists in evaluating the slope a . The time-scaling is supposed constant over the whole occurrence, since a varying time-scaling induces audible distortions not acceptable to the listener. All the segments thus share the same a value.

The *point-slope* of a pair of points (x_i, y_i) and (x_j, y_j) is defined as follows:

$$a_{i,j} = \frac{y_j - y_i}{x_j - x_i} \quad (2)$$

We evaluate the distribution of $a_{i,j}$, over the pairs i, j complying with the constraint $1s < x_2 - x_1 < D$, where D stands for the item duration. The complexity of this operation is linear $O(n)$. The lower bound is set because both coordinates are dis-

crete (as shown on the zoom provided Figure 3), which makes the point-slope less precise as the points get closer.

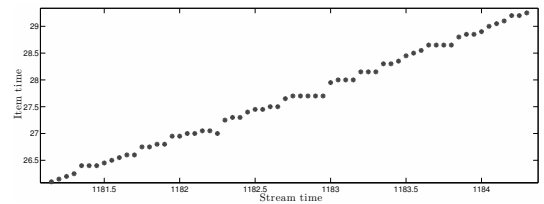


Figure 3: *Illustration of the discrete distribution of the points in the timestamp sequences.*

A histogram distribution is computed between the values 0.8 and 1.2² to identify the expected maximum peak on the a value. The computation of the point-slope indeed strongly amplifies the effect of a real line on the distribution, since aligned points all

²considered as large bounds for reasonable (i.e. not too audible) time factors.

contribute to the same bin, whereas unaligned points contribute to different bins. Figure 4 illustrates this on the case of points shared between two lines of same slope. Even in this case, one sees that the pairs from different lines induce different slopes, and will not create local maxima in the distribution.

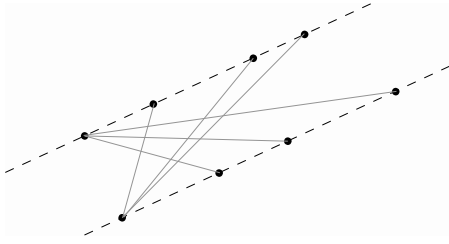


Figure 4: Unaligned points, even from aligned sets, do not contribute to local maxima in the distribution of slopes.

However there is in fact a "resonance" of the slopes in the very close vicinity of 1 ($|a - 1| < 0.0005$) that we don't explain. In order to avoid this accidental maximum, the distribution is set to zero in this very small interval. Even if the real maximum is precisely at 1, it will spread outside this interval and will be detected. Figure 5 shows an example of the slope distribution, where the time-factor peak is clearly located.

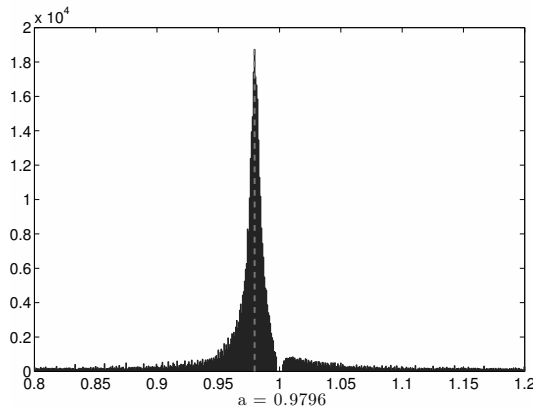


Figure 5: Example of histogram estimated distribution of the point-slopes. The slope a (i.e. the time-factor) is estimated from the salient peak position (indicated by the dotted line).

4.3. Offset estimation

We then estimate the *point-offsets* defined as follows:

$$b_i = y_i - a x_i. \quad (3)$$

Since all the lines share the same slope, then b_i is constant for all the points (x_i, y_i) on a same line. By estimating the distribution of the point-offsets, disjoint segments from the same line are gathered in the same bin, whereas segments from parallel lines (in the case of insertions or cuts) are separated.

Contrary to the time factor distribution, the search scope for the b values cannot be easily bounded. Moreover, in the case of noisy timestamp sequences, the local maxima are more spread than

the peak observed on the time-factor distribution. Instead of histograms, we thus use kernel density estimation [15] (with a gaussian kernel) to get a smoother distribution and gather neighboring peaks³. The latter, given a specific number of bins, automatically estimates the optimal bandwidth value for the gaussian kernel. The resulting distribution is then divided by its 90% percentile, in order to fix an absolute threshold (empirically set to 10) for peak detection.

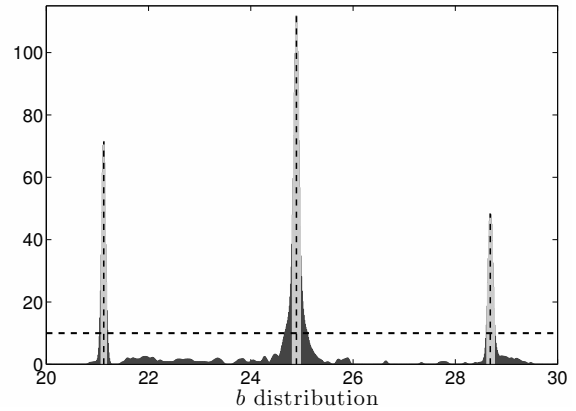


Figure 6: Distribution of the offset values ($y_i - a x_i$), evaluated with gaussian kernel density estimation. Here, three line offset values (indicated by the vertical dotted lines) are detected above the threshold (the horizontal dotted line). The light areas indicate the range of offsets associated with the peaks.

The peak values above the threshold are iteratively selected. At each step, the distribution is set to zero in the surroundings of the peak. The kernel bandwidth estimation induces the automatic determination of the 3 dB bandwidth of the peak. Finally, all the points (x_i, y_i) , with their offset inside the 3 dB cut-off interval, are associated to the selected b value, i.e. to a particular line in the timestamps plane.

Figure 6 shows the result of the offset estimation process applied to the timestamp sequence shown in Figure 2(f). Three peaks are shown, that correspond to the three last segments observed on Figure 2(f) (the fourth peak is outside the scope of the figure). Figure 7 shows the results of this operation on the timestamp sequence. Each gray shade represents the points associated with one of the segments.

In the case of no peak detection, the occurrence is discarded and considered as an erroneous annotation.

4.4. Segment estimation

The distribution of the points associated with the segments can be more noisy than on Figure 7, and needs post-processing. Figure 8 shows an example of result with erroneous points. Each ordinate (represented with a different shade) shows a binary signal that indicates the association (or not) of the points to the segment. In this case the segments 3 and 4 are to be discarded, and segments 1 and 2 show a few accidental points outside their boundaries. In fact any point can be wrongly associated with a line, if it is close

³In particular, we use the very fast and efficient implementation provided by Botev [16], but this is not essential here.

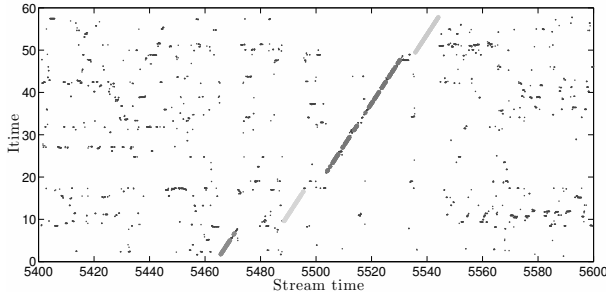


Figure 7: Each gray scale depicts the collection of points associated with one of the segment offsets detected from the timestamp sequence of Figure 2 (f).

to it, even if it is far outside the segment boundaries. These are discarded by applying a median filter, with a sliding window of 10 samples, on each segment binary signal. Then, the whole area between the first and the last point of each segment is assigned to it. Finally segments shorter than 5 s are discarded. The dotted boxes on the figure indicate the result of this post-processing, i.e. the estimation of the segment boundaries.

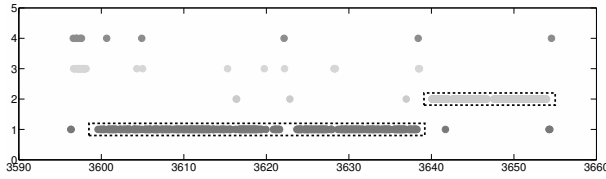


Figure 8: Each ordinate value corresponds to an offset value, and shows a binary signal indicating the point associated to it. The dotted boxes indicate the boundaries of the segments detected after post-processing. The offset values 3 and 4 are discarded here.

When detecting repetitions of the item (as in Figures 2(a) and (d)), only the most exact occurrence of the item is of interest, the other repetitions are accidental. A simple way to discard these repetitions is to compute the stream time distance between the segments: if a segment s_2 is shorter than s_1 , then the segment s_2 is discarded if $|x_{s_1} - x_{s_2}| > 30$ s, where $x_s = -\frac{b_s}{a}$ defines the abscissa of the intersection of the line s with the zero-ordinate axis.

4.5. Estimation of the temporal characteristics

The result of the process so far is a list of S segments; each segment s is characterized by its offset b_s and its boundaries $[x_{start}^s; x_{end}^s]$. The segments are sorted by ascending start boundaries. The slope a is common to all segments.

- TimeFactor is equal to the slope a :
TimeFactor = a .
- ItemTime is equal to the abscissa of the intersection of the first line with the zero-ordinate axis:
ItemTime = $-\frac{b_1}{a}$.
- StartTime equals the start boundary of the first segment:
StartTime = x_{start}^1 .
- EndTime equals the end boundary of the last segment:
EndTime = x_{end}^S .

- Successive segments with $b_{s+1} < b_s$ correspond to an insertion in the *stream* signal:
InsertTime (in the stream) = x_{end}^s ,
InsertDuration (in the stream) = $\frac{b_{s+1} - b_s}{a}$.
- Successive segments with $b_{s+1} > b_s$ correspond to an insertion in the *item* signal:
InsertTime (in the item) = $a(x_{end}^s - \text{ItemTime})$,
InsertDuration (in the item) = $b_{s+1} - b_s$.

Figure 9 illustrates the distinction on insertions. In order to cut the inserted chunks and synchronize both signals, the time factor is used differently when cutting an insertion in the stream (a) or in the item (b). The black line represents the two consecutive segments, and the gray segment represents the synchronized position for the second segment, after correction.

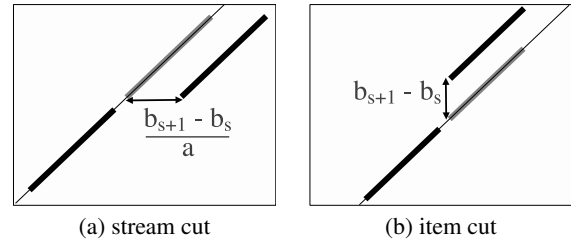


Figure 9: Illustration of the correction of an insertion detected between two consecutive segments. The estimate of the insertion duration is indicated.

5. EVALUATION

The full evaluation of such a process can only be done by human-checking all the occurrences that have been verified and aligned, as well as the occurrences discarded.

We have thus limited the evaluation to a subset of 100 occurrences randomly extracted from the Quero corpus annotations of the 2010 campaign of evaluation for audio identification. Some of the training items of the corpus were delivered by Yacast in several edit versions. Among the occurrences, we have then deliberately introduced 30 errors of edit version, in order to verify that edit mismatches are correctly identified as annotation errors. Item mismatch is supposed much more easy to detect than edit mismatch, and is thus not tested here.

After their automatic verification and alignment, the 100 occurrences were human-checked with the help of a small tool we developed prior to this contribution, in order to perform this synchronization manually. The tool interface (developed in Matlab) is shown Figure 10. The user can adjust the characteristics presented earlier (ItemTime, TimeFactor, etc.) and play the item and the aligned stream occurrence simultaneously to check the alignment. The software is meant to be used with headphones, since the item is played on the left channel and the occurrence on the right channel. The full description of the software is not relevant here.

After several hours of annotation we have concluded that the perception of a slight phase shift of d seconds is very consistent:

- $d \approx 0$: When sounds are perfectly simultaneous, one sound is heard and located in the middle of the head.
- $|d| < 0.03$: In a scope of about 30 ms, we still hear one sound, but the latter moves on the side when $|d|$ grows.

Figure 10: Graphical User Interface of the annotation tool developed for the manual alignment and verification of occurrences.

- $0.03 < |d| < 0.07$: Between 30 ms and 70 ms, we start hearing two different sounds on sharp onsets (e.g. percussive sounds or some consonants on the singing voice).
 - $|d| > 0.07$: Above 70 ms, we hear two different sounds.
- These empirical observations corroborate the results in the literature on spacial hearing perception [17].

We have thus limited the precision of the parameters to 0.01 s in the annotation tool, which is still notably heard (through the "perceived" position of the sound in the head) when approaching perfect synchronization.

Using the tool, we have corrected the result of our process to reach the best synchronization possible. Figure 11 shows the distribution of the corrections applied to the ItemTime parameter. Most of the correction amplitudes do not exceed 40 ms. The mean amplitude of the corrections equals 25 ms, which is an expected order of magnitude since the step size between the fingerprint codes was set to 50 ms. Some corrections, though reach higher values, up to 90ms. This is explained by the fact that a slight error on the time-factor can induce a much larger difference on time offsets, especially at the beginning of the occurrence, where the correction was applied.

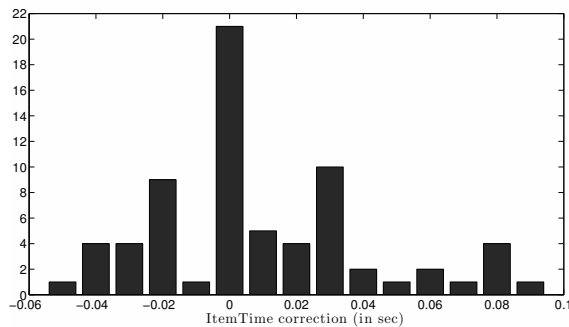


Figure 11: Histogram distribution of the corrections manually set to the ItemTime estimated with our method.

The 30 occurrences with edit mismatch were all detected as such, except one where the singing voice part is common to both edits. The detection of erroneous annotation is thus very efficient and reliable.

Finally, we have verified the TimeFactor estimation, by check-

ing that the two sounds are still perfectly synchronized after 20 seconds of signal. Only 4 occurrences over the 70 correct ones (about 6%) were slightly offbeat, the rest remains aligned.

Nevertheless, the best demonstration is still to actually hear the sounds. Several audio samples of the item and stream occurrence (before and after the automatic correction), as well as the stereo mix of the two, can be found on the following webpage: <http://www.mathieuramona.com/wp/data/align>.

6. CONCLUSION AND PERSPECTIVE

We have proposed here an original variant of the fingerprinting scheme, called item-restricted fingerprinting, that is associated with a segment detection method to estimate the temporal distortions between an item occurrence signal and the original item signal. The high precision of the parameters estimation allows the compensation of the temporal distortions and the perfect synchronization of the item and the occurrence.

This method has been used to verify and correct approximative annotations for audio fingerprinting. The short evaluation shows that the incorrect annotation detection works almost perfectly, even on different edits of the same musical track. The estimation of the temporal characteristic proves very precise and on most on the items, the perfect synchronization of the item and stream signal is confirmed perceptually, after compensating the temporal distortions.

This contribution offers many applications. In the field of audio fingerprinting, the alignment of the occurrences allows to reproduce a part of the evaluation protocols generally applied to synthetic alterations of items. Moreover, the precise estimation of the time-factor enables controlled studies on robustness to time-scaling on real-world audio data. The problem of signal alignment answered in this paper can probably be extended to other fields of research in audio processing.

Short-term perspectives would concern the remaining flaws of the algorithm. The peak near the value 1 in the distribution of the point-slopes deserves a proper explanation and should be answered more reliably. The correction of the insertions is also problematic. The duration is correctly estimated through the offset values, but the position is not precise enough, and results in local asynchrony between the signal. Proposing a proper scheme for the detection and the correction of missing occurrences in a fingerprint evaluation corpus is another long-term perspective.

7. ACKNOWLEDGMENTS

This work was partly supported by the "Quaero" Program funded by Oseo French State agency for innovation.

8. REFERENCES

- [1] Cyril Joder, Slim Essid, and Gaël Richard, "Hidden discrete tempo model: a tempo-aware timing model for audio-to-score alignment," in *Proc. ICASSP '11*, May 22-27 2011, pp. 397–400.
- [2] Arshia Cont, "A coupled duration-focused architecture for realtime music to score alignment," *IEEE Trans. on Pattern*

Analysis and Machine Intelligence, vol. 32, no. 6, pp. 974–987, June 2010.

- [3] Meinard Müller, Peter Grosche, and Frans Wiering, “Automated analysis of performance variations in folk song recordings,” in *Proc. ACM International Conference on Multimedia Information Retrieval (MIR '10)*, Philadelphia, Pennsylvania, USA, March 29-31 2010, pp. 247–256.
- [4] Meinard Müller, Frank Kurth, and Michael Clausen, “Chroma-based statistical audio features for audio matching,” in *Proc. IEEE Workshop on Applications of Signal Processing (WASPAA)*, New Paltz, New York, USA, October 2005, pp. 275–278.
- [5] Michael Casey, Christophe Rhodes, and Malcolm Slaney, “Analysis of minimum distances in high-dimensional musical spaces,” *IEEE Trans. on Audio, Speech and Language Processing*, vol. 16, no. 5, pp. 1015–1028, July 2008.
- [6] Mathieu Ramona, Sébastien Fenet, Raphaël Blouet, Hervé Bredin, Thomas Fillon, and Geoffroy Peeters, “Audio fingerprinting: a public evaluation framework based on a broadcast scenario,” *submitted to Special Issue on Event Recognition*, 2011.
- [7] Pedro Cano, Eloi Battle, Harald Mayer, and Helmut Neuschmied, “Robust sound modeling for song detection in broadcast audio,” in *Proc. AES Convention 112th*, 10-13 mai 2002, pp. 1–7.
- [8] Avery Li-Chun Wang, “An industrial-strength audio search algorithm,” in *Proc. ISMIR '03*, 2003.
- [9] Eugene Weinstein and Pedro Moreno, “Music identification with weighted finite-state transducers,” in *Proc. ICASSP '07*, April 15-20 2007, vol. 2, pp. 689–692.
- [10] Jaap Haitsma and Ton Kalker, “A highly robust audio fingerprinting system,” in *Proc. ISMIR '02*, 13-17 octobre 2002.
- [11] Xavier Rodet, Laurent Worms, and Geoffroy Peeters, “Brevet FT R&D/03376: Procédé de caractérisation d’un signal sonore - Patent 20050163325 Method for characterizing a sound signal,” *brevet international*, July 2003.
- [12] Mathieu Ramona and Geoffroy Peeters, “Audio identification based on spectral modeling of bark-bands energy and synchronisation through onset detection,” in *Proc. ICASSP*, May 22-27 2011, pp. 477–480.
- [13] Jean-Julien Aucouturier and François Pachet, “A scale-free distribution of false positives for a large class of audio similarity measures,” *Pattern Recognition*, vol. 41, no. 1, pp. 272–284, 2008.
- [14] Richard O. Duda and Peter E. Hart, “Use of the hough transformation to detect lines and curves in pictures,” *Communications of the ACM*, vol. 15, pp. 11–15, January 1972.
- [15] Emanuel Parzen, “On estimation of a probability density function and mode,” *Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.
- [16] Z. I. Botev, “A novel nonparametric density estimator,” *Tech. Rep.*, The University of Queensland, 2006.
- [17] Jens Blauert, *Spatial Hearing: The Psychophysics of Human Sounds Localization*, The MIT Press, revised edition edition, October 2 1996.

MULTI-PROBE HISTOGRAMS: A MID-LEVEL HARMONIC FEATURE FOR MUSIC STRUCTURE SEGMENTATION

Florian Kaiser, Thomas Sikora

Communication Systems Group
Technische Universität Berlin
Berlin, Germany

kaiser@nue.tu-berlin.de, sikora@nue.tu-berlin.de

ABSTRACT

The use of mid-level audio features has recently shown promising perspectives for music content description tasks. In this paper we investigate the use of a mid-level harmony-related descriptor for the task of music structure analysis. The idea behind the descriptor is to concatenate the chroma sequence of a musical segment in a Multi Probe Histogram (MPH). Probing local pitch intervals within the chroma sequence, we show that such an abstraction of the audio signal is a good descriptor of the tonal hierarchy that is consequent to a given key or melodic line. It is thus a powerful feature for describing harmonic and melodic progressions within a music piece. After introducing the histograms' computation and enlightening their relation to harmony, we use such a description for the task of music structure segmentation and provide preliminary results that show very encouraging perspectives.

1. INTRODUCTION

Music structure analysis aims at drawing the temporal map of a music piece by extracting its constitutive structural parts. In classical music, such structural forms usually correspond to the first and second movements, development, exposition and so on. In popular music, common structural segments are often referred to as intro, verse, chorus and outro. As a front-end processing for many challenging applications such as content-based information retrieval and browsing, summarization and thumbnailing, the task of music structural analysis has gained an increasing interest in the Music Information Retrieval community.

Most approaches for this task aim at identifying repetitive patterns or segments of homogeneous acoustical information in low-level audio descriptors. While perceptual studies show that sound properties such as timbre and harmony allow to discriminate sections within a music piece in most western music, MFCC and Chroma vectors are often chosen as low-level audio features. Different strategies were proposed to detect structural boundaries and classify sections in the features distributions. A popular approach consists in embedding the audio features in an audio self-similarity matrix [1] [2]. The comparison of the feature vectors in a pairwise manner enlightens repetitive and/or homogeneous segments within the audio data. Structure is then derived using k-means clustering or HMM. Other approaches directly apply clustering techniques to the features distribution. A comparative study of most recent algorithms can be found in [3].

A limitation of using low-level descriptors for the description of a music piece is that acoustical information within musical sections is highly inhomogeneous. Therefore the extraction time-

scale of the features, which is rather short, does not always yield a robust description of sections. Mid-level features or dynamic features were thus recently introduced to account for the temporal evolution of the feature vectors. In [4], authors model the temporal evolution of the spectral shape over a fixed time duration window. Varying the window size, authors can derive similarity matrices that either relate to short-term or long-term structures. In [5], Dynamic Texture is applied to the audio to model timbral and rhythmical properties. An alternative solution was proposed in [6] introducing a contextual distance that considers sequences of frames in order to enlighten repetitive patterns in the feature vectors.

We propose in this paper to use a mid-level harmony-related audio feature to describe musical structures, by means of a concatenation of mid-term chroma sequences in Multi-Probe Histograms (MPHs). MPHs were recently introduced in [7] for the task of scalable audio content retrieval and has not been used for musical audio content description yet. Probing local tone intervals within the sequence, such a representation allows to summarize the whole sequence by its dominant tone intervals and is thus an abstraction of its harmonic content.

After introducing the chroma representation of audio signals and the computation of MPHs, we discuss the musical interpretation of the resulting histograms. On the basis of musical knowledge and works on perception of tonal structures, we emphasize the harmony interpretation of MPH's. This is experimentally validated by analyzing the *Well-Tempered Clavier* books using this representation. Once the link between MPH and harmony has been established, MPHs are embedded in similarity matrices and used for the task of music structure segmentation. Evaluation of the system is reported at the end of the paper.

2. AUDIO SIGNAL REPRESENTATION

2.1. Chroma Features

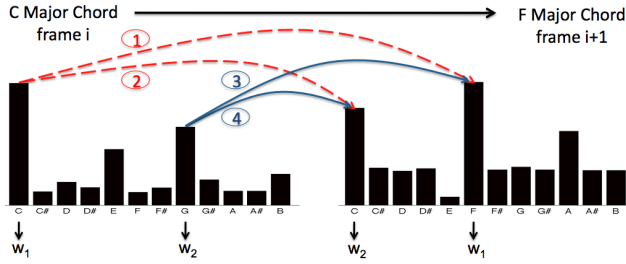
Chroma features are low-level audio descriptors that describe the pitch classes content of an audio data. Each coefficient of a chroma vector sums the signal's spectrum energy in sub-bands corresponding to one of the 12 pitch classes of the well-tempered scale. For the experiments in this paper, chroma features were extracted by means of the chroma toolbox¹. The analysis of chroma features thus enables to focus on the structure of harmonic-related content within a music piece.

¹<http://www.mpi-inf.mpg.de/~mmueller/chromatoolbox/>

2.2. Multi-Probe Histograms

Chroma features are extracted over neighboring windows of the audio signal. There is thus a strong correlation between adjacent feature frames, unless a strong transition occurs in the audio signal, such as a note change, and in that case, only the value of first dominant chroma bins changes. Multi-Probe Histograms are based on that observation and aim at characterizing these local transitions between dominant bins in chroma sequences by probing the pitch classes intervals between adjacent frames. A deepen description of MPH's computation is beyond the scope of this paper and can be found in [7]. Nevertheless, we will briefly introduce their computation by means of a simple example in order to understand how tonal structures can be reflected in the histograms.

We consider a transition from a C Major chord (frame i) to a F Major chord (frame $i+1$) and their corresponding chroma vectors:



From frame i to frame $i+1$, there are 12^2 possibilities for the maximum of energy to be transferred from a pitch class to an other. Each of these possible transitions defines a position in the Multi-Probe Histogram. In our example, the energy is logically transferred from the pitch classes of the major triad of a C to the major triad of an F. For the MPH computation, we consider for each adjacent frames of the sequence the transition between the two dominant pitch classes, i.e. C and G for the C Major chord and C and F for the F Major chord. As illustrated above, the transition from frame i to frame $i+1$ allows 4 possible transfers of energy between those pitch classes. For this iteration, bins of the MPH that will be allocated a new value are defined by these 4 transitions. Considering the transition from the tone C to the tone F, a bin position in the histogram is computed as follows :

$$p = b_C * 12 + b_F \quad (1)$$

with b_C and b_F the positions of the pitch classes C and F in the chroma vector, respectively 1 and 6. Furthermore, in each frame the first dominant bin is allocated the weight w_1 and the second the weight w_2 . C and F being the first dominant bins in our example, the added weight for our histogram bin for the transition from C to F is defined as:

$$w = w_1 + w_1 \quad (2)$$

meaning that the histogram is added the value w at its bin p . The operation is then repeated for the remaining 3 pitch classes transitions, and iterated over the remaining frames of the chroma sequence. Note that the actual values of major pitch classes in the chroma vectors do not influence the histogram's value. Only the distance or interval between tones does. Of course, one can increase the number of chroma bins K to be considered from one frame to another, thus probing 2^K intervals at each time frame.

As illustrated in the remainder of this paper, MPH's can either be computed with the whole chroma sequence of an audio data, or just over a portion of the audio signal and used as a mid-level audio feature. Independently of the length of the chroma sequence, computed MPH's size does not vary and is thus composed of $12^2 = 144$ bins.

3. MPH AS A HARMONIC FEATURE

Tonality and harmony relate to the combination of pitches in the chord constructions and melodic progressions of a music composition. In this section, we investigate how a MPH abstraction of chroma sequences is related to harmony and tonality. We first shortly review works on tonal structure perception in music cognition research. By means of the *well-tempered clavier books* we then experimentally show how key distances are reflected using the Multi-Probe Histograms as an audio feature.

3.1. Tonal Structures

Works and studies on the perception of tone structures in the music cognition literature show a strong interdependency between tones, chords and key. By means of the probe tone experiment, i.e. people are asked to judge of the quality of a note in a given key context, Krumhansl estimated key profiles, or tonal hierarchies, that are produced by a given harmony. Moreover, she states in [8] that such tonal hierarchies generate a map of key distances that is the same of the chord distances, and is verified in the circle-of-fifth. This means that given a tonal context, the transition probabilities between chords and pitch classes of the well-tempered scale are not equally distributed in a music composition. Moreover, a study by Cohen in [9] on how a key becomes established in a music composition showed that the four notes of a music piece (in that case excerpts of the *Well-Tempered Clavier Books*) were sufficient for musically trained listeners to estimate the tonic of the key. This all suggests that a music composition and its tonal structure are characterized by a restricted set of discrete pitches and intervals. The tonal structure is even established in mid-term sections in music pieces. Considering local pitch intervals, MPH's are completely determined by the tonal hierarchy of a key context, and should therefore be a good feature, or abstraction, for describing the particular tonal structure of a music piece.

3.2. Experimental Validation

The Well-Tempered Clavier books consist of preludes and fugues composed in all 24 major and minor keys and are considered as a reference work on harmony. In order to experimentally validate the interpretation of Multi-Probe Histograms as harmonic features, we extracted MPHs on the 24 preludes of the *The Well-Tempered Clavier* books and measured the distance between the pieces comparing their corresponding histogram. The goal is to find whether or not the MPH based comparison of the pieces satisfies the map of key distances that is defined by the circle-of-fifth. Note that Cohen verified in [9] that consonance between Bach Preludes are consistent with the circle-of-fifth.

For each piece, the chroma vectors are extracted with a sampling rate of 10Hz and are concatenated in a MPH. A piece is thus modeled by a single 144 bins histogram. Distance between the pieces is measured by calculating the cosine distance between the histograms. The results are shown in Figure 1.

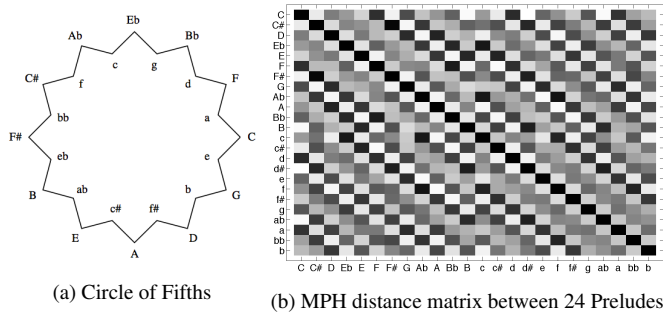


Figure 1: Circle of Fifths and MPH similarity between the 24 Preludes of the *Well-Tempered Clavier Books*

Computing the similarity between the pieces, we verify that keys that are close to each other in the circle-of-fifth, and that are thus consonant, also have a high MPH similarity. On the other hand, pieces composed in keys that are highly distant are highly dissimilar in their MPH representations. Whereas only measuring the similarity between the chroma sequences is not sufficient to yield that result, concatenating the sequences in MPHs thus allows to highlight the relevant tone intervals that are consequent to a given key. And in that sense it can be used as a good and computationally inexpensive harmonic descriptor of music.

4. STRUCTURE DETECTION

A Multi-Probe Histogram can be extracted over chroma sequences of variable sizes. It can thus be used as an abstraction of a whole music piece, but could also define a mid-level audio feature for music content description. In this section we investigate the use of MPH's as a mid-level audio feature for the task of music structural segmentation.

4.1. MPH as a mid-level audio feature

For the description of a music piece, we embed the MPH's as mid-level audio features in an audio self-similarity matrix. This means that instead of calculating the feature frame pairwise distance, as proposed in [1], each time instant is now modeled by the MPH calculated on a sequence of L frames that surrounds it. Thus including more contextual information in the measure of similarity, we intend to provide a more homogeneous description of structural parts. We can illustrate that with a simple example: let's consider a same single note, for example an A, that is played in two different sections of a music piece, and thus in two different melodic, and eventually tonal, contexts. The pairwise similarity between the feature frames extracted over these two notes will be maximal and rise confusion, whereas if one introduces contextual information, awareness of the past- and forthcoming tonal structure is considered and the similarity between the two time instants is reduced.

We show a concrete example in Figure 2. The audio excerpt is a 30 seconds excerpt of Chopin's *Mazurka, Op. 63, No. 3* in which a tonality change from a B flat minor to a C Sharp minor occurs. In Figure 2.a, the standard similarity matrix as proposed in [1] is computed on the chroma features. In Figure 2.b, we compute our proposed similarity matrix with MPH's computed over chroma sequences of length $L = 50$ frames, which corresponds to 5 seconds.

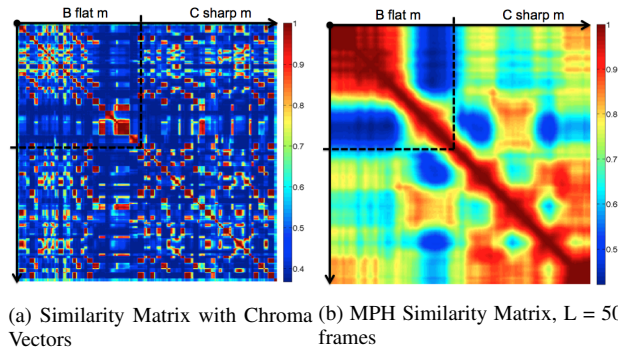


Figure 2: Similarity matrices computed over a portion of the *Mazurka, Op. 63, No. 3*. Transition between B flat minor and C Sharp minor.

The scales of B flat minor and C Sharp minor contain similar pitch classes. There is thus a high similarity in the chroma vectors and it is not clear from the chroma similarity matrix when in the music piece the tonality change occurs. But introduction of contextual information by means of the MPHs considerably reduces the similarity between the two sections. Moreover, each section tends to be more represented as a block of high similarity, satisfying the definition of a state representation of structure as introduced in [10]. For comparison, similarity matrices were also generated using the mid-term mean-value of Chroma vectors as features. While self-similarity within sections is also strengthened, high confusion between the two sections remains. We thus hope that the MPH representation will robustly enhanced the description of music pieces.

4.2. Segmentation and Structure Clustering

The Segmentation step aims at estimating the potential boundaries between the structural parts of a music piece. We use for that purpose the audio novelty approach as described in [11]. This method has already shown good performances for the task of structure segmentation.

We use the MPH enhanced similarity matrices as input for the structure clustering algorithm described in [12]. The algorithm is based on a nonnegative matrix factorization (NMF) of similarity matrices that separates musical sections in the matrix. The approach tends to work better when sections are displayed as blocks of high similarity in the matrix. This is referred to in the literature as the state representation of structure. Computing the standard similarity matrix on the low-level descriptors, structure is however rarely displayed in that manner. As shown above, introducing contextual information with the MPHs strengthens such a state representation and we therefore hope to improve the performance of the structure segmentation using our MPH based similarity matrix.

5. EVALUATION

5.1. Evaluation set-up and evaluation metrics

In order to compare our approach to the state of the art, we run the evaluation on the *TUT Beatles*² data set that consists of 174

²<http://www.cs.tut.fi/sgn/arg/paulus/structure.html>

songs from The Beatles. There is no ideal performance measure of music structure analysis algorithms. In fact musical structures being highly hierarchical, it is hard to find a match between the hierarchy-level of the annotation and the estimated structure. Nevertheless a compromise has been found in using the pairwise precision (P), recall (R) and F-measure (F), and the over- and under-segmentation scores (So and Su) introduced in [13].

The results are compared with the system in [14] that won the MIREX³ music structure segmentation evaluation task in 2009 for the same dataset, and with the same clustering algorithm we use [12], but ran on standard chroma similarity matrices.

5.2. Results

The evaluation is reported in Table 1.

	MPH based Similarity Matrix	Standard Matrix Ref [12]	Ref [14]
F	63.3%	60.8%	60.0 %
P	59.3%	61.5%	56.1%
R	72.4%	64.6%	71.0%
So	68.3%	61%	73.9%
Su	58.8%	59.9%	61.7%

Table 1: Evaluation of the proposed approach and comparison with the state-of-the-art

The general increase in the F-measure is of 3% in comparison with the reference systems. While the algorithm seems to behave in a similar manner as in [14] (comparable Precision and Recall rates), the nature of the segmentation changes using MPHs instead of raw chroma vectors for the similarity matrix computation. Indeed, introduction of the MPH increases the Recall rate of 8% in comparison with [12] with a reasonable loss in Precision (2%). This means that our approach deals better with over-segmentation issues. Over-segmentation is indeed often a problem in structure segmentation because of the inner structure of musical sections. While this structure is reflected in the estimated segmentation, it doesn't match the hierarchy level of the annotated structure.

It is also to be noted that our approach reflects structure in the harmonic progression of the songs. However, in this database, many structural information is also contained in the instrumentation changes. It would therefore be appropriate in further work to study the impact of MPH with mono-instrumental recordings.

6. CONCLUSION

In this paper we showed that concatenating chroma sequences in Multi-Probe Histograms is efficient for describing tonal and harmonic properties of sounds. Varying the length of the studied chroma sequences, MPHs can either be utilized as global descriptors of music pieces, or as a mid-level feature for music content description. The first evaluation of its application to the task of structure segmentation shows very promising results. It is however important to keep in mind that evaluation methods for the task of structure segmentation are still under active discussions, and the performance measures do not reflect all aspects of the relevancy of an estimated segmentation. Indeed, there is a lack of accuracy

in the definition of musical structures and annotation procedures. Further work will include the evaluation of the approach on a large mono-instrumental classical music database. Thus focusing on the harmonic aspects of structure, with no eventual confusion introduced by instrumentation changes, one could then run a deeper investigation on the benefits of using MPHs for the task of music structure analysis.

7. ACKNOWLEDGMENT

This work was supported by the European Commission under contract FP7-21644 PetaMedia.

8. REFERENCES

- [1] Jonathan Foote, "Visualizing music and audio using self-similarity," in *ACM Multimedia (1)*, 1999, pp. 77–80.
- [2] Geoffroy Peeters, "Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach," in *ISMIR*, 2007.
- [3] Jouni Paulus, Meinard Müller, and Anssi Klapuri, "Audio-based music structure analysis," in *ISMIR*, 2010.
- [4] Geoffroy Peeters, "Toward automatic music audio summary generation from signal analysis," in *ISMIR*, 2002, pp. 94–100.
- [5] Luke Barrington, Antoni B. Chan, and Gert Lanckriet, "Modeling music as a dynamic texture," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, 2010.
- [6] Meinard Müller and Frank Kurth, "Enhancing similarity matrices for music audio analysis," in *Proc. IEEE ICASSP*, 2006.
- [7] Y. Yu, M. Crucianu, V. Oria, and E. Damiani, "Combining multi-probe histogram and order-statistics based lsh for scalable audio content retrieval," in *ACM Multimedia*, 2010.
- [8] Carol L. Krumhansl and Roger N. Shepard, "Quantification of the hierarchy of tonal functions within a diatonic context," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 5, no. 4, pp. 579–594, 1979.
- [9] Annabel J. Cohen, "Tonality and perception: Musical scales primed by excerpts from the well-tempered clavier of j.s.bach," *Psychological Research*, vol. 53, no. 4, pp. 305–314, 1991.
- [10] Geoffroy Peeters, "Deriving musical structures from signal analysis for music audio summary generation: "sequence" and "state" approach," in *CMMR*, 2003.
- [11] Jonathan Foote, "Automatic audio segmentation using a measure of audio novelty," in *ICME*, 2000, p. 452.
- [12] Florian Kaiser and Thomas Sikora, "Music structure discovery in popular music using non-negative matrix factorization," in *ISMIR*, 2010.
- [13] Hanna M. Lukashevich, "Towards quantitative measures of evaluating song segmentation," in *ISMIRSMIR*, 2008.
- [14] Matthias Mauch, Katy Noland, and Simon Dixon, "Using musical structure to enhance automatic chord transcription," in *ISMIR*, 2009.

³http://www.music-ir.org/mirex/wiki/2009:Music_Structure_Segmentation_Results

PRODUCTION EFFECT: AUDIO FEATURES FOR RECORDING TECHNIQUES DESCRIPTION AND DECADE PREDICTION

Damien Tardieu, Emmanuel Deruty, Christophe Charbuillet and Geoffroy Peeters*

STMS Lab IRCAM - CNRS - UPMC

1 place Igor Stravinsky

75004 Paris, France

dtardieu, deruty, charbuillet, peeters (at) ircam.fr

ABSTRACT

In this paper we address the problem of the description of music production techniques from the audio signal. Over the past decades sound engineering techniques have changed drastically. New recording technologies, extensive use of compressors and limiters or new stereo techniques have deeply modified the sound of records. We propose three features to describe these evolutions in music production. They are based on the dynamic range of the signal, energy difference between channels and phase spread between channels. We measure the relevance of these features on a task of automatic classification of Pop/Rock songs into decades. In the context of Music Information Retrieval this kind of description could be very useful to better describe the content of a song or to assess the similarity between songs.

1. INTRODUCTION

Recent popular music makes an exhaustive use of studio-based technology. Creative use of the recording studio, referred to as production, exerts a huge influence on the musical content [1]. Sonic aspects of music, as brought by studio technologies, are even considered by some authors to be at the top of the hierarchy of pertinence in contemporary popular music analysis [2]. They can be perceived as more important than rhythm and even than pitch.

Studio techniques may concern many aspects of the musical content. Equalizers modify spectral content, reverberation bring customizable acoustics to the recording, pitch-shifters like Antares Autotune¹ can transform vocals to a point where it becomes the trademark of a song [3]. Double and multiple tracking techniques allow the construction of heavily contrapuntal and spatialized parts from a single original sound source or musician [4]. Dynamic processing used in audio mastering weights so heavily on music perception that it spawns public debate [5].

Studio practices are heavily dependent on equipment: equalizers and dynamic compressors require electronic components, pitch-shifting is impossible to perform without digital processing and recordings have to be made on media whose performance are highly variable across the musical periods. This leads to the hypothesis that some sonic aspects in recorded music are specific to a given period of time.

In the Music Information Retrieval field, this aspect has received few attention. The first work which could be related to production is the Audio Signal Quality Description Scheme in MPEG-7 Audio Amendment 1 [6]. This standard includes a set of audio

features describing the characteristics of the support, considered as a transmission channel of a music track: description of Back-GroundSoundLevel, RelativeDelay, Balance, Bandwidth. In [7], Tzanetakis uses the Avendano's Panning Index [8] to classify production styles (and then production time). Kim [9] and Scaringella [10] study the effect of remastering on the spectrum of the songs. Their interest in remastering comes from a question that was more debated, the so-called "album effect". This refers to the fact that machine learning algorithms for automatic music classification or music similarity estimation may learn characteristics of the album production instead of general properties such as genre and then be over fitted. Identifying this album effect is still an open problem but we believe that some production aspects do not belong to the album effect and may characterize the period of a song or even its genre. It is thus important to characterize the production effect that is independent of album and relates to more general attributes of a song. In this aim we propose three features describing some aspects of the production of a song. The first feature relates to the temporal variation of the signal amplitude and is described in section 2, the second and the third, detailed in section 3 describe the use of stereo. To assess the accuracy of these features and how they relate to a production period, we use them in a task of automatic classification of songs in decades in section 4. We finally conclude in section 5.

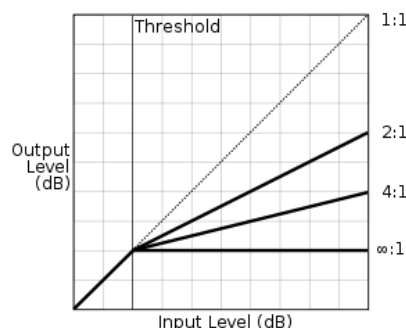


Figure 1: Relationship between input and output level in a compressor for a fixed threshold and various ratios.

* Part of this work was made as an independant consultant

¹<http://www.antarestech.com/>

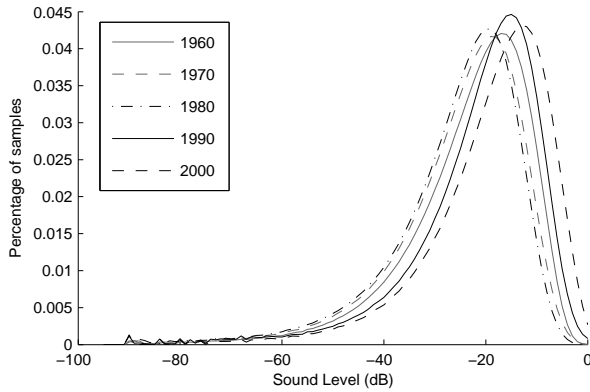


Figure 2: Mean decibel amplitude histogram for five decades from 1960 to 2000

2. COMPRESSION AND LIMITING

2.1. A growing use of compressors and limiters over the years

The first techniques we study are compression and limiting. Their aim is to alter the amplitude of the signal in order to reduce its dynamic range (ie. the ratio between the loudest and the weakest parts of the signal's power). They can be used to deal with technical limitations of the recording system, or to improve the audibility of the signal for aesthetic reasons. A compressor applies a non-linear transformation to the sound level across time (see Fig. 1). It applies a negative gain to the signal whenever the amplitude exceeds a user-set threshold. Another way to deal with dynamic compression is to consider that one applies an input gain to the signal, which increases its power, while the signal's peaks must not get over a given threshold under in any circumstance. This is the principle of limiting. Intensive usage of limiting results in signals with many samples very close to 0 dB Full Scale (the maximum possible level on digital media). From the beginning of the 90s, this technique has been increasingly used to make songs sound "louder" while peaking at the same level. Each music recording company wanting to make records that sound "louder" than the ones from the competitor, this degenerated into a so-called "loudness war" (see for instance [11]). To describe these effects we propose a feature based on the amplitude of the signal in dB FS (Decibel Full-Scale).

2.2. Signal description of compression and limiting effects

2.2.1. Dynamic histogram

This feature corresponds to the histogram of the peak normalized signal level represented in dB. Let $s(n)$ be the audio signal with $s(t) \in [-1, 1]$.

$$s_{dB}(n) = 20 * \log_{10}(|x(n)|) \quad (1)$$

The bins of the histogram are 1dB wide and the centers go from -95.5 dB to -5.5dB. These values are chosen considering the 96 dB dynamic of a 16-bit signal. The histogram is normalized to represent percentage values.

We use the signal amplitude instead of any energy estimate to be able to precisely detect the effect of limiting. Indeed, both limiters

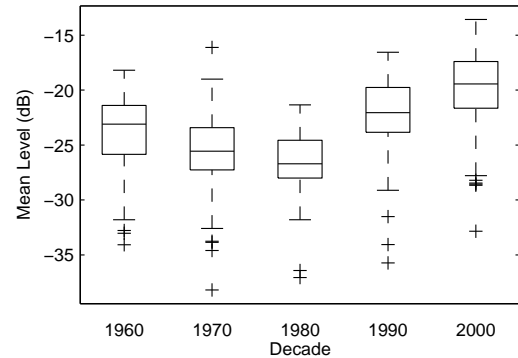


Figure 3: Mean of the signal amplitude absolute value in dB for five different decades. The center horizontal line represents the median. The middle two horizontal lines represent the upper and lower limits of the inter-quartile range. The outer whiskers represent the highest and lowest values that are not outliers. Outliers, represented by '+' signs, are values that are more than 1.5 times the inter-quartile range.

and compressors apply a gain directly on the signal. As a result, the effect of the limiter, as it has been used recently, will be the presence of many samples with an amplitude very close to 0 dB. If we were using an energy estimate, such as RMS, these peak values would be smoothed and then less visible.

Fig. 2 shows the mean amplitude histogram computed on 1042 Pop/Rock songs (see section 4 for details) for five different decades ranging from 1960 to 2010. First we notice the progressive displacement of the histogram toward the right, ie. toward high sound level value, from the 80s to the 00s. This is typically the effect of a higher compression rate over decades. Looking at Fig. 3 representing the mean of the signal amplitude absolute value in dB for the five decades, we can confirm the increase of the sound level from the 80s to the 00s, but we also see that this value decreases from the 60s to the 80s. This diminution of the mean sound level can be explained by the increasing bandwidth of the recording media from less than 75dB in the 60s [12] to the 96 dB of the audio CD. Indeed, if the bandwidth increases and the peak value stays constant, the mean decreases. The second noticeable observation on these histograms appears on the high sound level bins, particularly in the [-1dB,0dB] bin. Indeed we see that the height of these bins increase with the decade. This is an effect of the intensive use of limiters, and justifies the use of amplitude instead of energy estimation. This effect is more visible on Fig. 4 that shows the percentage of samples between -1 and 0 dB (ie. the height of the rightest bin of the histogram) for the five decades. We see that this value does not vary much in the three first decades and starts growing in the nineties to reach a top value in the 00s.

2.2.2. Summary features

To obtain a more compact representation, we also compute the four first moment of the distribution of s_{dB} , ie. the mean, the variance, the skewness and the kurtosis, as well as the median and inter-quartile range. In the following we call these features, together with the histogram bin amplitude, the *Dynamic Features*. A higher compression rate should be materialized by a higher mean

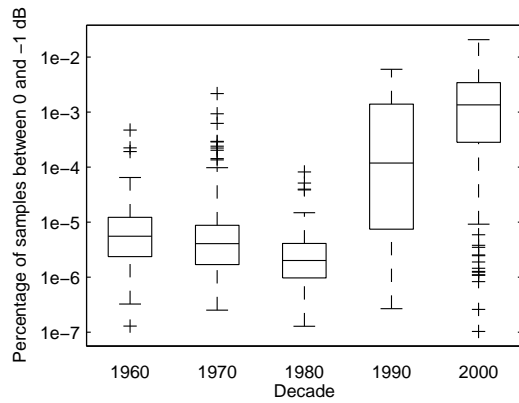


Figure 4: Percentage of samples between -1 and 0 dB for five different decades.

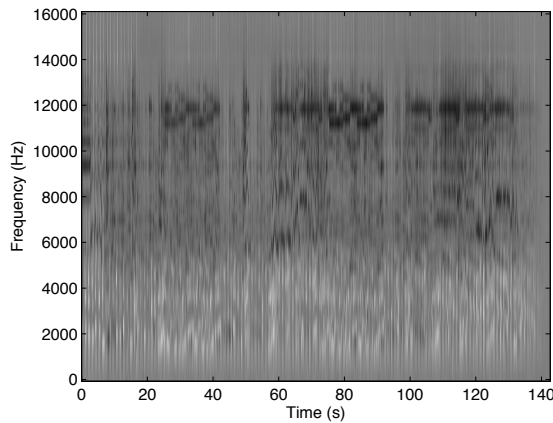


Figure 5: Cochleagram difference for the *While My Guitar Gently Weeps* from *The Beatles*. Color ranges from -0.3 (white) representing right channel to 0.3 (black) representing left channel.

or median and also by a lower skewness (the mass of the distribution is concentrated on the high values). Fig. 3 shows the mean of the distribution over decades. The observation is the same as on the histogram, showing an increasing use of compression from the 90s.

3. STEREO AND PANNING

The second group of techniques that we study relates to the differences that are observed between the left and right channels of a stereo recording. This panel of techniques results in a variety of signals, that can range from mono ones, for which there is no difference between the two channels, to complex stereo images produced by using amplitude and phase differences. We present two measures that intent to describe the differences between the two channels.

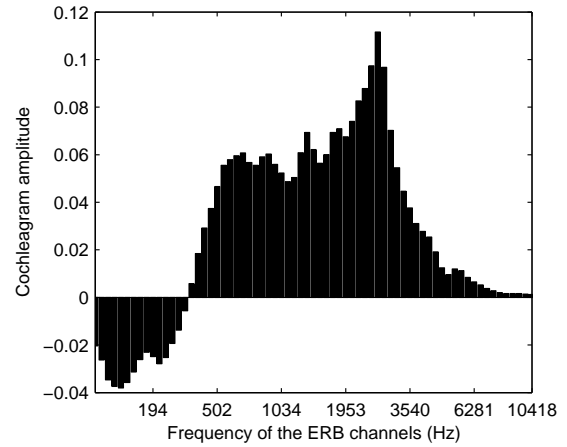


Figure 6: Mean of cochleagram difference for the song *While My Guitar Gently Weeps* from *The Beatles*. Negative values indicate right channel, positive values indicate left channel.

3.1. Amplitude panning

3.1.1. Cochleagram differences

Amplitude panning consists in distributing the sound of each sources on each channel. Avendano [8] proposes a method to measure the differences between left and right channels. He computes a normalized similarity measure between left and right channel spectrograms. We use a slightly different measure based on channel cochleagrams. The cochleagram represents the excitation pattern of the basilar membrane. We use this method to obtain a more perceptually meaningful representation of the sound. The cochleagram is computed using a gammatone filterbank whose center frequencies follows the ERB scale [13]. The ERB scale is computed as follows:

$$ERB_n = 21.4 \log_{10}(0.00437f + 1); \quad (2)$$

where f represent the frequency.

We use a filterbank of 70 filters with frequency centers between 30 Hz and 11025 Hz. To measure the spectral difference between channels over time, we compute the difference between both channel cochleagram. We call this representation *Cochleagram Difference* (CD). Fig. 5 shows the Cochleagram Difference of the song *While My Guitar Gently Weeps* from *The Beatles*. We can clearly see the guitar and the organ (in black) between 500 Hz and 3 kHz that are almost fully panned on the left. In the low frequency we notice (in white) the bass and the drums that are much louder on the right channel. The remaining green color is mainly due to voices that are in the center.

3.1.2. Summary features

To summarize the information contained in this representation we use four features that we will call *Amplitude Stereo Features* (ASF) in the following.

- The global mean over frequency and time of the absolute value. This feature indicate the global amount of panning in the song,
- The standard deviation over frequency of the mean over time of the absolute value. This feature is an indication of the amount of panning variation across time,
- The mean over time which measure the mean panning across frequencies,
- The mean over time of the absolute value which gives the same indication but ignoring the panning direction (left/right),
- The standard deviation over time.

As an illustration, Fig. 6 shows the mean over time of the cochleagram difference for the same song as in Fig. 5. This figure shows that, over the song, bass frequencies are panned on the right (indicated by negative values), while medium and high frequencies are panned on the left (indicated by positive values).

3.2. Phase stereo

In the last two decades, sound engineers have been broadly using mixing techniques based on slight differences between the left and right channel that give a sense of “wideness” to the sources. We will group these techniques under the designation of “phase stereo” as opposed to “amplitude stereo”, of which panning is an example. There exist at least three of these techniques. The simplest one is based on an inversion of phase between the two channels. Another one is based on a single original track, that is being panned as it is on one channel, and panned with a short delay (between 10 and 30ms) on the other channel. A third one, sometimes called “double-tracking”, consists in recording at least twice the same musical phrase played on the same instrument and to pan each take on a different channel. This method is widely used by heavy metal producers on guitar parts, in order to provide an impression of a “huge” guitar sound. Such techniques are made easy to implement by the precision of track synchronization brought by reliable multi-track recorders, as well as the abundance of available tracks provided by digital recording systems. As a consequence of the use of these mixing techniques, recordings with very few panning can still give a sense of space. To describe these effects we propose a new representation inspired by the phase meters used by sound engineers.

3.2.1. Spectral Stereo Phase Spread (SSPS)

We denote by $s_L(n)$ and $s_R(n)$ the left and right channel of a stereo audio signal over sample n . The common tools used in music production to analyze the stereo de-phasing of an audio signal is named the “phase-meter”. It displays over a 2D representations the values $y(n) = s_L(n) - s_R(n)$ (on the ordinate) and $x(n) = s_L(n) + s_R(n)$ (on the abscissa). When the channels L and R are “in phase”, $y(n)$ cancels, when they are in phase opposition, $x(n)$ cancels. This is illustrated in Fig. 7. We therefore use the following σ_{LR} to measure the spread of the audio signal due to de-phasing

$$\sigma_{LR} = \frac{\sigma(s_L(n) - s_R(n))}{\sigma(s_L(n) + s_R(n))} \quad (3)$$

where $\sigma(x)$ denotes the standard deviation of the values x .

As for the Cochleagram Difference (which measures stereo spread in frequency due to amplitude panning), we propose a formulation of the stereo phase spread in the frequency domain. The

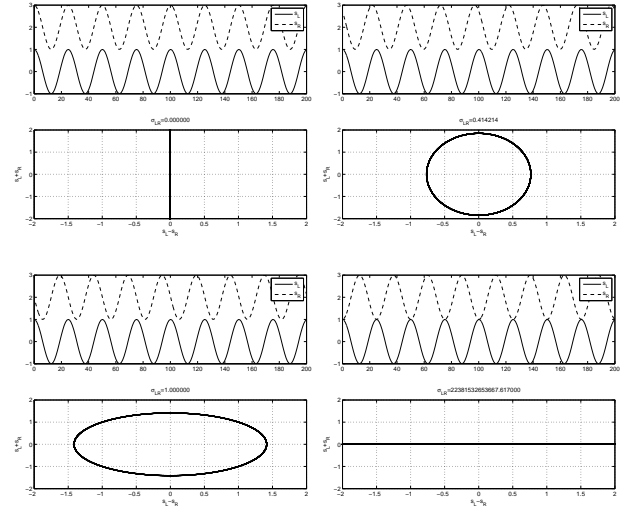


Figure 7: Representation of $s_L(n);s_R(n)$ (top parts) and $y(n);x(n)$ for various case of de-phasing between left and right channels. [top-left]: $\phi = 0$, [top-right]: $\phi = \pi/4$, [bottom-left]: $\phi = \pi/2$ and [bottom-right]: $\phi = \pi$

goal is to obtain a spectral location of the use of de-phasing techniques. For this a Short Time Fourier Transform analysis is first performed using a Blackman window of length 40ms with a 20ms hop size. We denote by $S_L(f_k, m)$ and $S_R(f_k, m)$ the respective short time Fourier complex spectrum at frame m and frequency f_k . The phase components, $\Phi_L(f_k, m)$ and $\Phi_R(f_k, m)$ represents the phase of each cosinusoidal component at frequency f_k and at the beginning of the frame. The phase components $\Phi_L(f_k, m)$ and $\Phi_R(f_k, m)$ over frame m can therefore be considered as an equivalent of $s_L(n)$ and $s_R(n)$. We can therefore compute the same measures $Y(f_k, m) = S'_L(f_k, m) - S'_R(f_k, m)$ and $X(f_k, m) = S'_L(f_k, m) + S'_R(f_k, m)$ using

$$\begin{aligned} S'_L(f_k, m) &= \cos(2\pi f_k / sr + \Phi_L(f_k, m)) \\ S'_R(f_k, m) &= \cos(2\pi f_k / sr + \Phi_R(f_k, m)) \end{aligned} \quad (4)$$

$$\sigma_{LR}(k) = \frac{\sigma(S'_L(f_k, m) - S'_R(f_k, m))}{\sigma(S'_L(f_k, m) + S'_R(f_k, m))} \quad (5)$$

In order to derive a perceptual measure from $\sigma_{LR}(k)$, we group the values over frequencies f_k into ERB bands.

$$\sigma_{LR}(b) = \sum_{f_k \in \{B\}_k} \sigma_{LR}(k) \quad (6)$$

where $\{B\}_k$ denotes the set of frequency of the b^{th} ERB bands. A further refinement is to weight each value of $\sigma_{LR}(k)$ by the amplitude of the corresponding frequency bin f_k

$$\sigma'_{LR}(b) = \sum_{f_k \in \{B\}_k} A(k) \sigma_{LR}(k) \quad (7)$$

where $A(k)$ is the mean of the contribution of the modulus (amplitude spectrum) $|S_L(f_k, m)|$ and $|S_R(f_k, m)|$.

In Fig. 8, we illustrate the computation of $S'_L(f_k, m) - S'_L(f_k, m)$ and $S'_L(f_k, m) + S'_L(f_k, m)$ for five frequency bands and de-phasing of $\phi = 0, \phi = \pi, \phi = \pi/2, \phi = \pi/4$ and $\phi = 0$ in each band.

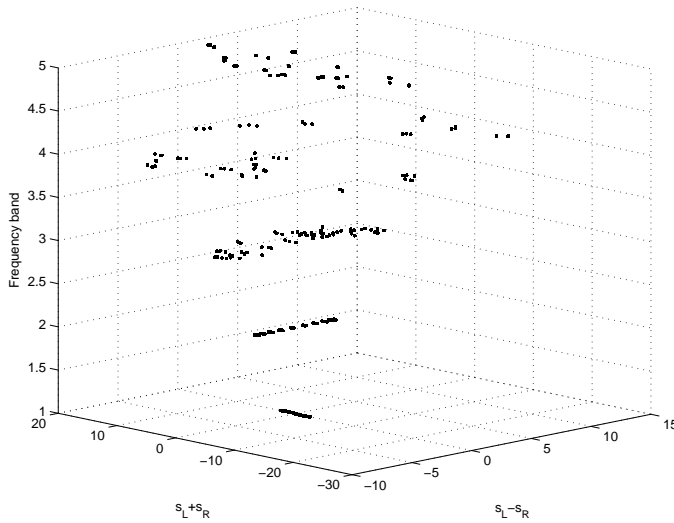


Figure 8: Computation of $\sigma_{LR}(k)$ in the frequency domain.

Fig 9 shows the CD (on the top) and the SSPS (on the bottom) of the song *Gangsta's Paradise* by *Coolio*. Compared to the previous *Beatles'* song, this song presents very few amplitude panning as shown by the almost uniform green color of the CD. In contrast the SSPS shows some very strong variations. The lighter areas of the SSPS corresponds to points in time and frequency where the phase difference between channels is higher. These segments correspond to the entrance of the choir which as been mixed using the *double tracking* technique.

3.2.2. Summary features

To summarize the information contained in SSPS we use four features that we will call *Phase Stereo Features* (PSF) in the following.

- The global mean over frequency and time,
- The standard deviation over frequency of the mean over time,
- The mean over time,
- The standard deviation over time.

4. CLASSIFYING SONGS INTO DECADES

As a proof of concept of our features we propose to automatically classify songs into decades. Since the proposed features are designed to describe production characteristics of the records, and since these characteristics have changed over time, our features should allow to guess the period of production. This kind of classification could be very interesting for measuring the similarity between songs or for automatically generating playlists.

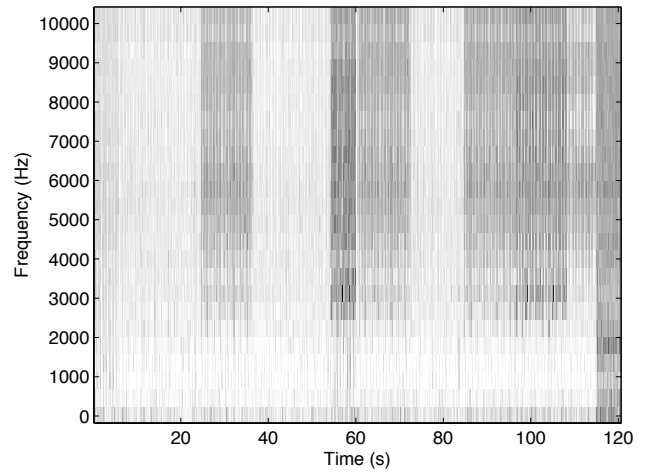
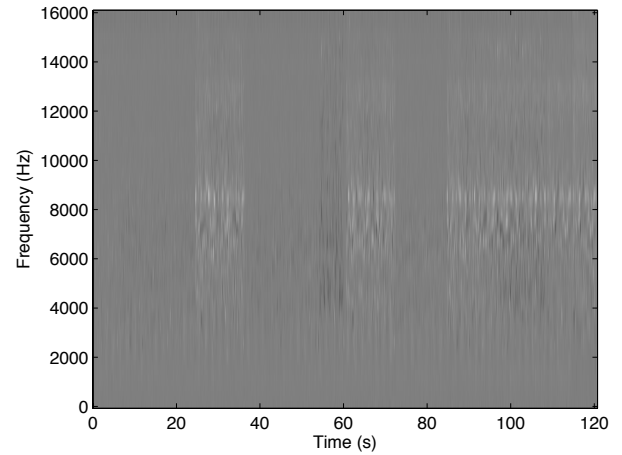


Figure 9: Cochleagram Difference (Top) and Spectral Stereo Phase Spread (Bottom) for the song *Gangsta's Paradise* by *Coolio*. In the Cochleagram Difference, Color ranges from -.3 (white) representing right channel to .3 (black) representing left channel. In the Spectral Stereo Phase Spread lighter colors represent higher phase spread.

4.1. Sound set

We use a set of 1980 Pop/Rock songs by 181 different artists. The set contains 396 songs for each decade. The year were obtained from a metadata database. The set is divided into a train set of 1042 songs and a test set of 938 songs. To avoid over-fitting of the models due to the album effect, the train and test sets contains different artists.

4.2. Classification method

As a classifier, we use support vector machines (SVM) with a Gaussian radial basis function kernel. We set $\gamma = 1/d$ [14] where d is the dimension of the feature set and $C = 1$. The implementation is the one of LIBSVM [15]. To make a multi-class classifier from the 2-class SVM we use the one versus all method. We train a classifier for each class versus all the remaining classes. To make a decision we compare the posterior probabilities provided

Features				Performance
DF	ASF	PSF	MFCC	Accuracy
		×		0,39
	×			0,46
			×	0,47
×				0,47
	×	×		0,51
×	×	×		0,61
×	×	×	×	0,64

Table 1: Classification accuracy for various feature combinations. DF=Dynamic Features, ASF=Amplitude Stereo Features, PSF=Phase Stereo Features, MFCC=Mel Frequency Cepstral Coefficients

real class	Classified as					recall
	1960	1970	1980	1990	2000	
1960	125	23	5	1	0	0,81
1970	28	111	78	12	7	0,47
1980	1	30	152	5	1	0,80
1990	16	36	43	125	24	0,51
2000	5	13	11	29	161	0,74
precision	0,71	0,52	0,53	0,73	0,83	

Figure 10: Confusion matrix for the classification with all the features

by LIBSVM and affect the class with the highest probability to the incoming data.

We compare the results of the proposed features either separately or grouped. For comparison purposes we added the Mel Frequency Cepstral Coefficients (MFCC) that are widely used features for spectral envelope description.

4.3. Results

Tab. 1 shows the classification results for various feature combinations. First, we see that every features carry information about decade, the best one being the dynamic features with an accuracy of 0.47. An interesting observation is that the two kind of stereo features (amplitude and phase) perform better when used together (.51) than separately (respectively .39 and .45), showing that they carry different kind of information. When all the features are used in conjunction we obtain a score of .64. Tab. 10 shows the confusion matrix of this last case. As expected, the main confusions occurs between adjacent decades. The 00s obtain the best recognition rate (.83) followed by the 90s and 60s (resp. .73 and .71). Confusion occurs more often between 70s and 80s.

5. CONCLUSION

In this paper, we presented three innovative audio features to describe the characteristics of the music production effect. These features are related to dynamic range and stereo mixing. Dynamic features pointed out the increasing use of compressors and limiters across decades. Stereo features were shown to be able to characterize both amplitude panning and phase stereo. The relevance of the features was tested in a task of automatic decade classification of music tracks. An accuracy of 60% on a five decade task was

reached using our features. While such classification can be useful for automatic song tagging or for music similarity, it could be interesting to try regression methods to estimate more precisely the within decade period of production. Also, since the production techniques can vary across genres, further research should focus on possible variations of our features across genres.

6. ACKNOWLEDGEMENT

This work was supported by French Oseo Project QUAERO.

7. REFERENCES

- [1] Virgil Moorefield, *The Producer as Composer: Shaping the Sounds of Popular Music*, The MIT Press, 2005.
- [2] François Delalande, *Le son des musiques entre technologie et esthétique*, INA-Buchet/Chastel, Paris, 2001.
- [3] Sue Sillitoe, "Recording Cher's 'Believe'," *Sound on Sound magazine*, Feb. 1999.
- [4] Emmanuel Deruty, "Archetypal vocal setups in studio-based popular music," in *EIMAS*, Rio de Janeiro, Brazil, 2010.
- [5] Etan Smith, "Even Heavy-Metal Fans Complain That Today's Music Is Too Loud!!!," *The Wall Street Journal*, Sept. 29, 2008.
- [6] MPEG-7-Audio-Amendment-1, "Information Technology - Multimedia Content Description Interface - Part 4: Audio," ISO/IEC FDIS 15938-4/A1, ISO/IEC JTC 1/SC 29.
- [7] George Tzanetakis, Randy Jones, and Kirk Mc Nally, "Stereo panning features for classifying recording production style," in *International Symposium on Music Information Retrieval*, Vienna, Austria, 2007.
- [8] Carlos Avendano, "Frequency-domain source identification and manipulation in stereo mixes for enhancement, suppression and re-panning applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003, pp. 55–58, IEEE.
- [9] Youngmoo E. Kim, Donald S. Williamson, and Sridhar Pilli, "Towards quantifying the album effect in artist identification," in *ISMIR*, Victoria, Canada, 2006, pp. 393–394.
- [10] Nicolas Scaringella, *On the Design of Audio Features Robust to the Album-Effect for Music Information Retrieval*, Ph.D. thesis, 2009.
- [11] Sarah Jones, "Dynamics are Dead, Long Live Dynamics-Mastering Engineers Debate Music's Loudness Wars,," 2005.
- [12] John M. Eargle, *Handbook of Recording Engineering*, Springer, 1996.
- [13] Brian R. Glasberg and Brian C. J. Moore, "Derivation of auditory filter shapes from notched-noise data," *Hearing Research*, vol. 47, no. 1-2, pp. 103–138, Aug. 1990.
- [14] Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik, "Extracting Support Data for a Given Task," in *Proceedings of the First International Conference on Knowledge Discovery & Data Mining*, 1995, pp. 252–257, AAAI Press.
- [15] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1—27:27, 2011.

Satellite Workshop 1 - Versatile Sound Models for Interaction in Audio-Graphic Virtual Environments: Control of Audio-graphic Sound Synthesis - *Roland Cahen, Diemo Schwarz, Hui Ding*

The use of 3D interactive virtual environments is becoming more widespread in areas such as games, architecture, urbanism, information visualization and sonification, interactive artistic digital media, serious games, gamification. The limitations in sound generation in existing environments are increasingly obvious with current requirements. This workshop will look at recent advances and future prospects in sound modeling, representation, transformation and synthesis for interactive audio-graphic scene design. Several approaches to extending sound generation in 3D virtual environments have been developed in recent years, such as sampling, modal synthesis, additive synthesis, corpus based synthesis, granular synthesis, description based synthesis, physical modeling... These techniques can be quite different in their methods and results, but may also become complementary towards the common goal of versatile and understandable virtual scenes, in order to cover a wide range of object types and interactions between objects and with them. The purpose of this workshop is to sum up these different approaches, present current work in the field, and to discuss their differences, commonalities and complementarities.

Satellite Workshop 2 - Modalys, a physical synthesizer: more than twenty years of researches developments and musical uses - *Nicolas Ellis, Joel Bensoam, Jean Lochard, René Caussé*

In the early 90s, the software was initially created with the intent to serve as a virtual instrument maker workshop. The usages are now extending from the virtual reproduction of existing acoustic instruments to industrial prototyping. This diversification made necessary to rethink many parts of the software, from the core synthesiser to the numerous interfaces : textual (Lisp), Max/MSP (mlys), OpenMusic, Matlab.

Satellite Workshop 3 - From ASA to CASA, what does the C stand for anyway? - *Mathieu Lagrange, Luis Gustavo Martins*

Auditory Scene Analysis (ASA) is the process by which the human auditory system organizes sound into perceptually meaningful elements. Inspired by the seminal work of Al Bregman (1990) and other researchers in perception and cognition, early computational systems were built by engineers such as David Mellinger (1991) or Dan Ellis (1996). Strictly speaking, a CASA or a "machine listening" system is an computational system whose general architecture or key components design are motivated by facts taken from ASA. Though, ASA being a Gestaltist theory that focuses on the description and not on the explanation of the studied phenomenon, computational enthusiasts are left with a largely open field of investigation. Perhaps this lack of definition did not fit into the way we do research nowadays, since papers strictly tackling this issue are relatively scarce. Though, informal discussions with experts in the sound and musical audio processing areas confirm that making sense of strongly polyphonic signals is a fundamental problem that is interesting both from the methodological and application point of views. Consequently, we (organisers of this workshop) believe that there are fundamental questions that need to be raised and discussed in order to better pave the way of research in this field. Among others, those questions are: From ASA to CASA: only insights? Is the knowledge transfer from ASA to CASA only qualitative? Are there other approaches in scientific fields such as biology, cognition, etc. that are also potentially meaningful for building powerful computational systems? What is CASA? Is CASA a goal in itself? Can it be decomposed into well defined tasks? Is CASA worth pursuing? What are the major locks in contemporary CASA? How does it relates to other sound processing areas such as Blind Source Separation (BSS) or Music Information Retrieval (MIR)? This workshop aims at bringing to the audience some background and new topics on ASA and CASA . Those questions will then be raised and discussed with the help of the invited speakers

Index of Authors

- Ahmad, Wasim, [409](#)
Anaïk, Olivero, [123](#)
Aramaki, Mitsuko, [387](#)
Avizienis, Rimas, [313](#)
- Böck, Sebastian, [135](#)
Balazs, Peter, [107](#)
Battenberg, Eric, [313](#)
Beller, Greg, [277](#)
Bello, Juan, [171](#)
Bilbao, Stefan, [337](#)
Bonada, Jordi, [73](#), [371](#)
Bouchara, Tifanie, [167](#)
- Canazza, Sergio, [177](#)
Caramiaux, Baptiste, [167](#)
Charbuillet, Christophe, [425](#), [441](#)
Chen, Yin-Lin, [63](#)
Cho, Taemin, [171](#)
Clifford, Alice, [1](#)
Contreras, Javier, [321](#)
- Dörfler, Monika, [23](#), [93](#)
Daudet, Laurent, [375](#), [393](#)
David, Bertrand, [393](#)
Degottex, Gilles, [277](#)
Dempwolf, Kristjan, [205](#), [257](#)
Depalle, Philippe, [81](#)
Derrien, Olivier, [387](#)
Deruty, Emmanuel, [441](#)
Dutoit, Thierry, [269](#)
- Esquef, Paulo, [401](#)
Evangelista, Gianpaolo, [345](#)
- Farner, Snorre, [277](#)
Fazekas, György, [119](#)
Fdili Alaoui, Sarah, [167](#)
Fink, Marco, [365](#)
Florian, Kaiser, [437](#)
Foher, Dominique, [213](#)
Fontana, Federico, [287](#)
Forsyth, Jon, [171](#)
Frank, Matthias, [307](#)
Freiberger, Karl, [185](#)
- Gómez, Emilia, [73](#)
Gamper, Hannes, [37](#)
Garcia, Francisco, [357](#)
Girin, Laurent, [353](#)
Gnann, Volker, [101](#)
- Grill, Thomas, [93](#)
- Hélie, Thomas, [45](#)
Hiipakka, Marko, [209](#)
Holighaus, Nicki, [93](#)
Holters, Martin, [31](#)
Huber, Stephan, [277](#)
- Inoguchi, Yasushi, [69](#)
Iwaya, Yukio, [69](#)
- Kang, Laewoo, [171](#)
Kenmochi, Hideki, [241](#)
Kereliuk, Corey, [81](#)
Klapuri, Anssi, [249](#)
Kleimola, Jari, [233](#)
Kondoz, Ahmet, [409](#)
Kröning, Oliver, [205](#)
Kraft, Sebastian, [301](#)
Kronland-Martin, Richard, [387](#)
- Lallemant, Ianis, [291](#)
Lanchantin, Pierre, [277](#)
Lazzarini, Victor, [115](#), [233](#)
Letz, Stéphane, [213](#)
Liao, Wei-Hsiang, [63](#), [141](#)
Liuni, Marco, [107](#)
- Macak, Jaromir, [59](#)
Maestre, Esteban, [357](#), [417](#)
Maller, Simon, [321](#)
Marchini, Marco, [417](#)
Marentakis, Georgios, [307](#)
McGovern, Stephen, [11](#)
Michon, Romain, [199](#), [361](#)
Moinet, Alexis, [269](#)
Musevic, Saso, [371](#)
- Norilo, Vesa, [217](#)
- Obin, Nicolas, [277](#)
Oksanen, Sami, [27](#)
Orlarey, Yann, [213](#)
Otani, Makoto, [69](#)
- Papiotis, Panagiotis, [417](#)
Parker, Julian, [37](#), [163](#)
Parseihian, Gaetan, [167](#)
Peeters, Geoffroy, [127](#), [277](#), [425](#), [429](#), [441](#)
Pekonen, Jussi, [19](#)
Pérez, Alfonso, [357](#), [417](#)
Perng, Chao-Yu Jack, [329](#)

Pihlajamäki, Tapani , [19](#)
Pinel, Jonathan, [353](#)
Pulkki, Ville, [209](#)

Röbel, Axel, [107](#), [141](#), [277](#), [321](#)
Rébillat, Marc, [167](#)
Régnier, Lise, [127](#)
Rabenstein, Rudolf, [365](#)
Ramona, Mathieu, [265](#), [429](#)
Ramos, German, [253](#)
Reiss, Josh, [1](#)
Richard, Gaël, [265](#)
Rigaud, François, [393](#)
Rodà, Antonio, [177](#)
Rodet, Xavier, [277](#)
Rossing, Thomas, [329](#)

Salamon, Justin, [73](#)
Salvati, Daniele, [177](#)
Sandler, Mark B., [119](#)
Sarver, Ryan, [249](#)
Sato, Yukinori, [69](#)
Schedl, Markus, [135](#)
Schimmel, Jiri, [59](#)
Schwarz, Diemo, [221](#), [291](#)
Serafin, Stefania, [53](#)
Siedenburger, Kai, [23](#)
Sikora, Thomas, [437](#)
Sirdey, Adrien, [387](#)
Smith, Julius, [199](#), [329](#), [361](#)

Sontacchi, Alois, [185](#), [307](#)
Spagnol, Simone, [209](#)
Spiertz, Martin, [101](#)
Sturm, Nicolas, [375](#)
Su, Alvin Wen-Yu, [63](#), [141](#)

Tardieu, Damien, [425](#), [441](#)
Tassart, Stéphan, [147](#)
Timoney, Joseph, [115](#), [233](#)
Turchet, Luca, [53](#)

Välimäki, Vesa, [19](#), [27](#), [37](#), [233](#)
Veaux, Christophe, [277](#)
Velasco, Gino Angelo, [93](#)
Villavicencio, Fernando, [241](#), [277](#)
Vincel, Leny, [357](#)
von dem Knesebeck, Adrian, [301](#)
Vorländer, Michael, [155](#)

Wang, Tien-Ming, [63](#)
Wefers, Frank, [155](#)
Wells, Jeremy, [191](#)
Welter, Guilherme, [401](#)
Wilmering, Thomas, [119](#)

Yeh, Chunghsin, [141](#)
Yiyu, Tan, [69](#)

Zölzer, Udo, [31](#), [205](#), [257](#), [301](#)
Zambon, Stefano, [287](#)
Zotter, Franz, [307](#)