ADAPTIVE HARMONIZATION AND PITCH CORRECTION OF POLYPHONIC AUDIO USING SPECTRAL CLUSTERING

Mathieu Lagrange, Graham Percival, and George Tzanetakis

Computer Science Dept.
University of Victoria, British Columbia
Canada
[lagrange, gperciva, gtzan]@uvic.ca

ABSTRACT

There are several well known harmonization and pitch correction techniques that can be applied to monophonic sound sources. They are based on automatic pitch detection and frequency shifting without time stretching. In many applications it is desired to apply such effects on the dominant melodic instrument of a polyphonic audio mixture. However, applying them directly to the mixture results in artifacts, and automatic pitch detection becomes unreliable. In this paper we describe how a dominant melody separation method based on spectral clustering of sinusoidal peaks can be used for adaptive harmonization and pitch correction in mono polyphonic audio mixtures. Motivating examples from a violin tutoring perspective as well as modifying the saxophone melody of an old jazz mono recording are presented.

1. INTRODUCTION

Pitch correction and harmonization are some of the most common digital audio manipulations. They are typically applied to recordings of monophonic sound sources such as the singing voice or melodic instruments. Currently if the original monophonic sound source is not available it is not possible to apply these effects on polyphonic audio mixtures. The main problem is that the required frequency shifting is also applied to the accompaniment, background music resulting in severe artifacts. In addition, pitch correction and harmonization are adaptive effects that rely on the output of an automatic pitch detection algorithm. In general pitch detection of the dominant melody in polyphonic audio is much harder than the monophonic case and in most cases unreliable.

In this paper we describe how a dominant melody separation algorithm inspired by ideas in Computational Auditory Scene Analysis (CASA) can be utilized for applying pitch corrections and harmonizations to the melody in polyphonic audio recordings. Unlike systems that use stereo panning information [1] our focus is mono recordings where there is a clear dominant melody such as a violin with piano accompaniment or saxophone and trumpet melodies in jazz recordings. The source separation algorithm is based on a sinusoidal representation and a spectral clustering technique is used to group the peaks of the dominant melody. Perceptually informed grouping cues such as amplitude/frequency proximity and harmonicity are utilized. Pitch correction and harmonization are straightforward to express using a sinusoidal representation by simple frequency scaling of the peaks. This assumes that the underlying fundamental frequency of the dominant melody is known. We utilize a pitch detection algorithm based on [2] and show that it works better using the separated signals.

The remainder of the paper is structured as follows. The dominant melody separation algorithm is described in section 2. Pitch correction and harmonization are described in section 3. Experiments with a violin tutoring system and altering the saxophone melody in a jazz recording are described in section 4. The paper ends with conclusions and directions for future work.

2. SOUND SOURCE FORMATION

Computational Auditory Scene Analysis (CASA) systems aim at identifying perceived sound sources (e.g. notes in the case of music recordings) and grouping them into auditory streams using psycho-acoustical cues [3]. However, as remarked in [4] the precedence rules and the relevance of each of those cues with respect to a given practical task is hard to assess. Our goal is to provide a flexible framework where these perceptual cues can be expressed in terms of similarity between time-frequency components. The identification task is then carried out by clustering components which are close in the similarity space. Therefore, the complexity of the algorithm is strongly related to the number of components considered. In this paper we use time-varying sinusoids as the underlying time-frequency representation.

2.1. Sinusoidal Modeling

Most CASA approaches consider auditory filterbanks and/or correlograms as their front-end [5]. In these approaches the number of time-frequency components is relatively small. However closely-spaced components within the same critical band are hard to separate. Other approaches [4, 6, 7] consider the Fourier Spectrum as their front-end. In these approaches, in order to obtain sufficient frequency resolution a large number of components is required. Components within the same frequency region can be pre-clustered together according to a stability criterion computed using statistics over the considered region. However, this approach has the drawback of introducing another clustering step, and opens the issue of choosing the right descriptors for those pre-clusters. Alternatively, a sinusoidal front-end is helpful to provide meaningful and precise information about the auditory scene while considering only a limited number of components, see Figure 1, and is the representation we consider in this work.

Sinusoidal modeling aims to represent a sound signal as a sum of sinusoids characterized by amplitudes, frequencies, and phases. A common approach is to segment the signal into successive frames of small duration so that the stationarity assumption is met. For each frame a set of sinusoidal components is estimated.

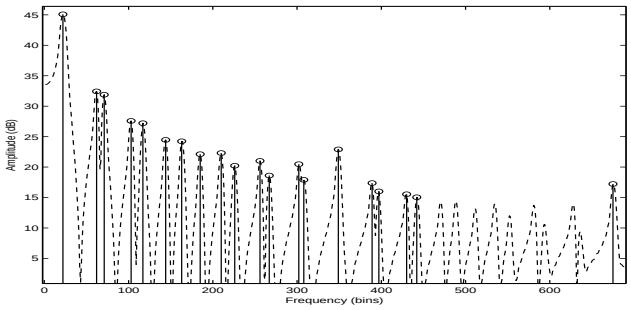

Figure 1: *Picking of 20 peaks in the spectrum of a mixture of two harmonic sources.*

The discrete signal $x^k(n)$ at frame index k is then modeled as follows:

$$x^k(n) \quad = \quad \sum_{l=1}^{L^k} a_l^k \cos\left(\frac{2\pi}{F_s} f_l^k \cdot n + \phi_l^k\right) \qquad (1)$$

where F_s is the sampling frequency and ϕ_l^k is the phase at the beginning of the frame of the l-th component of L^k sine waves. The f_l and a_l are the frequency and the amplitude of the l-th sine wave, respectively, both of which are considered constant within the frame. For each frame k, a set of sinusoidal parameters $\mathcal{S}^k = \{p_1^k, \cdots, p_{L^k}^k\}$ is estimated. The system parameters of this Short-Term Sinusoidal (STS) model $\mathcal{S}^k$ are the L^k triplets $p_l^k = \{f_l^k, a_l^k, \phi_l^k\}$, often called *peaks*.

These parameters can be efficiently estimated by picking some local maxima from a Short-Term Fourier Transform (STFT) with a frame size of 46ms and a hop size of 11ms. The precision of these estimates is further improved using phase-based frequency estimators which utilize the relationship between phases of successive frames [8]. Using this enhanced frequency estimate, the rough amplitude estimate provided by the magnitude of the local maximum is also corrected.

2.2. Spectral Clustering

In order to simultaneously optimize partial tracking and source formation, we construct a graph over the entire duration of the sound mixture of interest. Unlike approaches based on local information [9], we utilize the global normalized cut criterion to partition the graph (spectral clustering). This criterion as been successfully used for image and video segmentation [10]. In our perspective, each partition is a set of peaks that are grouped together such that the similarity within the partition is maximized and the dissimilarity between different partitions is maximized. By appropriately defining the similarity between peaks, a variety of perceptual grouping cues can be used.

To express such similarity, the edge weight connecting two peaks p_l^k and $p_{l'}^{k'}$ (k is the frame index and l is the peak index) may depend on the proximity of frequency, amplitude and harmonicity:

$$W(p_l^k, p_{l'}^{k'}) = W_f(p_l^k, p_{l'}^{k'}) \cdot W_a(p_l^k, p_{l'}^{k'}) \cdot W_h(p_l^k, p_{l'}^{k'}) \qquad (2)$$

where W_x are typically radial basis functions of distance among the two peaks in the x axis. For more details see [11, 12].

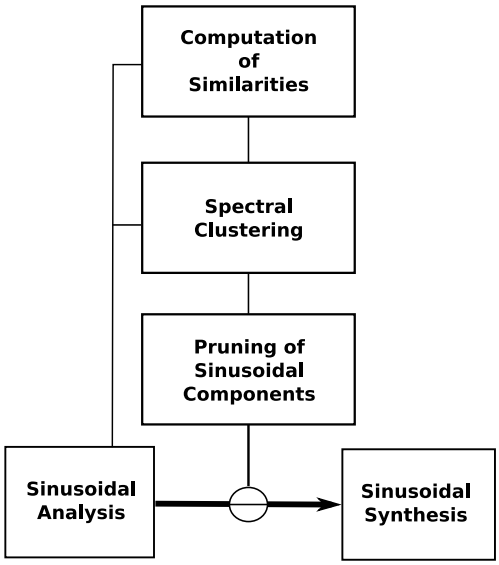

Figure 2: *Block-Diagram of the Dominant Melody Segregation Algorithm.*

Most existing approaches that apply the Ncut algorithm to audio [13, 7] consider the clustering of components over one analysis frame only. However, the time integration (partial tracking) is as important as the frequency one (source formation) and should be carried out at the same time. We therefore propose in [12] to consider the sinusoidal components extracted within a texture window of several spectral frames (20 in the experiments). Figure 2 shows a block diagram of the proposed separation scheme.

We considered a maximum of 40 sinusoids per frame. Those frames are 20 ms long. We chose to select two out of three clusters of peaks for each texture window to perform the resynthesis. The clusters with the highest average within similarity based on Equation 2 are selected. An example of separation is depicted in Figure 3. A bank of sinusoidal oscillators is used for resynthesis.

3. PITCH CORRECTION AND HARMONIZATION

Pitch correction and harmonization both rely on the ability to modify the perceived pitch of a sound without changing its duration in time. For the sounds we are interested in, the pitch directly corresponds to the fundamental frequency of the sound, therefore we do not differentiate between pitch and fundamental frequency. A sinusoidal representation is ideally suited for this purpose and therefore most systems for pitch modification and harmonization are based on the phasevocoder [14]. These effects work particularly well for signals with slowly varying harmonic structure that have small amounts of transients and background noise.

Using a sinusoidal representation, pitch shifting is achieved by scaling the frequencies of each peak in the representation. In the case of pitch correction we multiply the peaks corresponding to the dominant melody and do not retain the original peaks. For harmonization both the original and the multiplied peaks are retained for resynthesis and multiple scaling factors are used to form chords.

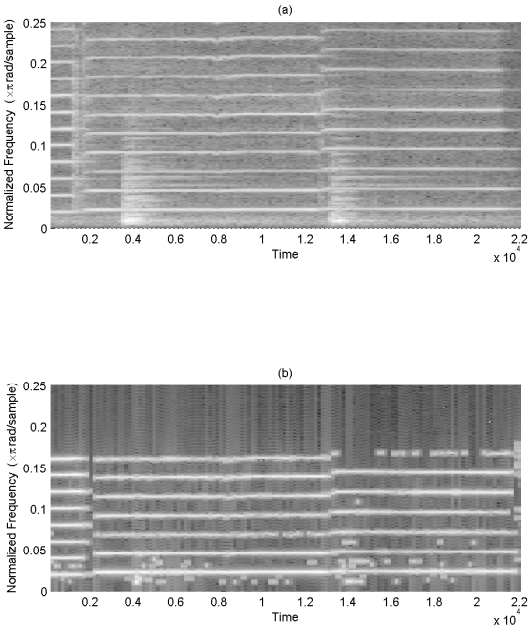

Figure 3: *Example of Segregation of the Violin (b) from a Violin accompanied by Piano.*

3.1. Pitch Detection

In order to know the required amount of frequency scaling it is necessary to know the underlying pitch of the sound during the frame under consideration. To automatically determine the pitch an implementation of the method described in [2] was used. The estimation of the pitch from a monophonic signal is a well studied area and robust methods exist, see [5] for a review.

We utilize the autocorrelation function to efficiently estimate the fundamental frequency ($f0$). For a time signal $s(n)$ that is stationary, the autocorrelation $r_s(\tau)$ as a function of the lag τ is defined as

$$r_s(\tau) = 1/N \sum_{j=t}^{t+N} s(t)s(t+\tau) \quad (3)$$

This function has a global maximum for $\tau = 0$. If there are also additional global maxima, the signal is called periodic and there exists a lag τ_0, the period, so that all these maxima are placed at the lags $n\tau_0$, for every integer n, with $r_s(n\tau_0) = r_s(0)$.

The inverse of the lag τ_0 provides an estimation of the fundamental frequency $f0$. The period is determined by scanning $r_t(\tau)$, starting at zero, and stopping at the first global maximum with non-zero abscissa. Quadratic interpolation is used to further improve the frequency estimation. In practical cases, the relative amplitude of those maxima may change and some others maxima may appear due to small aperiodicities of the signal. The issue is then to relevantly select which maximum corresponds to the $f0$ by considering several candidates under a plausible range and pick the one with the highest confidence, see [2, 15] for further references on the algorithm. To avoid picking harmonics as the fundamental frequency the amplitude of the peaks is weighted using the following equation:

$$r_s(\tau) = r(\tau_{max}) - OctaveCost^2 * log(MinPitch * \tau_{max}) \quad (4)$$

3.2. Note Segmentation

After the pitch contour has been generated we apply note segmentation to convert the frequency values in Hz to floating-point MIDI note numbers. The contour is scanned with a window size of 500ms. If the median frequency value is sufficiently far away from the previously detected notes (we found that 0.6 MIDI notes produced good results), the sample is flagged as a note boundary.

Once we have divided the audio into notes, we compare the pitches with the pitches given by a MIDI file. For each audio frame inside each note, we compute

$$frequency\_multiplier = \frac{midi2Hz(expected\_pitch)}{midi2Hz(detected\_pitch)} \quad (5)$$

It would be desirable to previously estimate the tuning of the performance. This way, if a different tuning standard (A = 442 Hz instead of 440 Hz, for example) is desired, we can easily change the mapping from Hz to floating-point values.

Currently, if a vibrato is present in the performance, the frequency contour is linearized within each note. Removing any micro-modulations of the pitch contour is desirable for correcting the performances of inexperienced students. However, this is not desirable for performances of skilled musicians who deliberately produce such modulations (vibrato, glissandi, ...).

3.3. Harmonization

Adding chords to a melody is a standard task in Computer Assisted Composition (CAC) [16], but this generally requires an analysis of the entire piece of music and is not suitable for online processing. In some cases we can use simple online algorithms to add chords. The easiest to automatically add harmonies is to have them precomputed by a human: "at 1.0 seconds, play a C chord. At 1.5 seconds, play a G7 chord, etc". This requires that we know exactly what pitch the user should be creating at every moment.

While this is unreasonable for an improvised performance of contemporary computer music, there are still situations where this constraint is quite valid. In music pedagogy, we often ask students to play their instrument in time with a metronome – in other words, to play specific notes at exactly the right time. This is also true of karaoke – since we are playing a vocal-less version of the music, we know what the user should be singing at every moment of the song. In these cases, adding pre-composed harmonies based purely on the elapsed time is an entirely appropriate method.

In this paper we use a simple and naive harmonization algorithm as a proof-of-concept. Simple harmonies may be added by examining only the current pitch. If we know that the user will be playing a simple melody in a specific key, we can calculate the scale degree of each note. "Scale degree" is the musical term for number of semitones within the octave. Adequate – although not particularly interesting – harmonies can be created by examining only the scale degree. For this paper, we used this simple method to demonstrate that harmonization is possible. More sophisticated approaches can easily be added to the system but are beyond the scope of this paper.

	violin solo	violin + piano	violin sep.
good intonation	97.9%	83.5%	93.1%
bad intonation	93.1%	79.9%	93.0%

Table 1: *Comparison of pitch recognition for separated and non-separated audio.*

4. EXPERIMENTS

In order to demonstrate how our proposed systems works we show examples from two application areas: pitch correction in music pedagogy and modifying the saxophone melody in monophonic jazz recordings. Audio examples can be found at `http://opihi.cs.uvic.ca/Dafx2007/`

Students learning instruments without fixed pitch (*i.e.* violin, cello, trombone) spend a significant amount of time concentrating on their intonation. Music teachers may demonstrate sections for the students, or encourage students to listen to recordings, but the sound of an expert playing an expensive musical instrument is quite different from the sound of a beginner playing a cheap musical instrument. Students may compare their sound to the expert's sound and become discouraged or distracted from the current goal of correct intonation. On the other hand, playing the correct melody using MIDI samples results in poor sound quality.

Using the student's own sound avoids these problems: the student hears music as it would sound if she played it with correct intonation. This eliminates any doubt (or possible excuses) for the student: the student listening to their pitch-corrected sound knows that she should – and *can* – produce exactly the same sounds. The audio examples are based on the following scenario: a violin student practicing with a tutoring system with piano accompaniment. A standard laptop microphone is used for acquisition and the recording is noisy. Various examples of pitch correction and harmonization for this scenario can be found on the webpage.

We also achieved significantly better pitch detection (and therefore better note segmentation) by separating the dominant melody from the original mixed audio. Table 1 shows our results; any pitch that was within 1.0 MIDI note of the real value was deemed to be "correct".

Another example application is the pitch correction of melodies in old mono recordings that are either live or for which the original master tapes of the individual instruments are not available. In the example provided on the webpage we modify the saxophone melody of a well known jazz standard by pitch shifting certain notes in the separated signal and then remixing with the residual. The result sounds quite good and to some extent the resynthesis artifacts are masked by the addition of the residual background. We are currently working on improving the resynthesis quality.

5. CONCLUSIONS

Dominant melody separation using a spectral clustering approach over a sinusoidal representation can be used for adaptive pitch correction and harmonization of polyphonic audio recordings.

Directions for future work include: incorporating more cues into the peak similarity calculation, improving resynthesis by reducing artifacts, more sophisticated harmonization, and adding the pitch correction functionality to an open source audio editing environment such as Audacity.

6. REFERENCES

[1] C. 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.

[2] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *Proceedings of the Institute of Phonetic Sciences, Amsterdam*, 1993, vol. 17, pp. 97–110.

[3] A.S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound*, MIT Press, 1990.

[4] E. Vincent, "Musical source separation using time-frequency priors," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 14(1), pp. 91–98, 2006.

[5] D. Wang and G. J. Brown, Eds., *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, Wiley, 2006.

[6] S.H. Srinivasan, "Auditory blobs," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.

[7] F.R. Bach and M. I. Jordan, "Blind one-microphone speech separation: A spectral learning approach," in *Proc. Neural Inf. Processing Systems (NIPS)*, Vancouver, Canada, 2004.

[8] M. Lagrange and S. Marchand, "Estimating the instantaneous frequency of sinusoidal components using phase-based methods," *to appear in the Journal of the Audio Engineering Society*, 2007.

[9] R.J. McAulay and T.F. Quatieri, "Speech analysis/synthesis based on sinusoidal representation," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 34(4), pp. 744–754, 1986.

[10] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22(8), pp. 888–905, 2000.

[11] M. Lagrange and G. Tzanetakis, "Sound source tracking and formation using normalized cuts," in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, USA, 2007.

[12] Mathieu Lagrange, Luis Gustavo Martins, Jennifer Murdoch, and George Tzanetakis, "Normalized cuts for singing voice separation and melody extraction," *submitted to the IEEE Trans. on Acoustics, Speech, and Signal Processing (Special Issue on Music Information Retrieval)*.

[13] S.H. Srinivasan and M. Kankanhalli, "Harmonicity and dynamics based audio separation," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

[14] J. Laroche and M. Dolson, "New phase-vocoder techniques for real-time pitch shifting, chorusing, harmonizing, and other exotic audio modifications," *J. Audio Eng. Soc*, vol. 47, no. 11, 1999.

[15] P. Boersma and D. Weenink, "Praat: doing phonetics by computer (version 4.5.06)," Retrieved December 13, 2006, from http://www.praat.org/.

[16] G. Assayag, "Computer assisted composition today," in *1st Symposium on Music and Computer, "Applications on Contemporar y Music Creation, Esthetic and Technical Aspects"*, 1998.