THE CATERPILLAR SYSTEM FOR DATA-DRIVEN CONCATENATIVE SOUND SYNTHESIS

Diemo Schwarz

Ircam – Centre Pompidou, Paris, France
`schwarz@ircam.fr`

ABSTRACT

Concatenative data-driven synthesis methods are gaining more interest for musical sound synthesis and effects. They are based on a large database of sounds and a unit selection algorithm which finds the units that match best a given sequence of target units. We describe related work and our CATERPILLAR synthesis system, focusing on recent new developments: the advantages of the addition of a relational SQL database, work on segmentation by alignment, the reformulation and extension of the unit selection algorithm using a constraint resolution approach, and new applications for musical and speech synthesis.

1. INTRODUCTION

Data-driven concatenative synthesis methods use a large database of source sounds, segmented into *units*, and a *unit selection* algorithm that finds the sequence of units that match best the sound or phrase to be synthesised, called the *target*. The selection is performed according to the *descriptors* of the units, which are characteristics extracted from the source sounds, or higher level descriptors attributed to them. The selected units can then be transformed to fully match the target specification, and are concatenated. However, if the database is sufficiently large, the probability is high that a matching unit will be found, so the need to apply transformations is reduced. The units can be *non-uniform*, i.e. they can comprise a sound snippet, an instrument note, up to a whole phrase.

Usual sound synthesis methods are based on a model of the sound signal. It is very difficult to build a model that would realistically generate all the fine details of the sound. Concatenative synthesis, on the contrary, by using actual recordings, preserves entirely the fine details of sound. For example, very naturally sounding transitions can be synthesized, since unit selection is aware of the context of the database units. In this *data-driven approach*, instead of supplying rules constructed by careful thinking as in a *rule based approach*, the rules are induced from the data itself. Findings in other domains, e.g. speech recognition, corroborate the general superiority of data-driven approaches.

In section 2, we briefly describe related work and then focus on our CATERPILLAR synthesis system [1] in section 3, and recent new developments and applications in section 4, before the conclusion in section 5.

2. RELATED WORK

Concatenative synthesis was first developed as a part of *text-to-speech* (TTS) synthesis systems. Data-driven non-uniform unit selection from large databases [2] was the key to augmenting the quality of the synthesised speech and the ease of adding new voices, and is used in virtually all new commercial and research systems, see for instance [3, 4, 5, 6].

Recently, the number of musical synthesis projects using ideas of concatenative data-driven synthesis began to rise sharply. All these approaches can be seen as a specialisation of the data-driven synthesis method proposed here, i.e. they could all be unified within the CATERPILLAR framework.

Plunderphonics [7] is John Oswald's artistic project consisting of songs made up from tens of thousands of snippets from a decade of pop songs, selected and assembled by hand.

Soundscapes [8] generates endless but never repeating soundscapes for installations by concatenating segments from a recording. It keeps the "texture" of the original sound file, avoiding discontinuities.

Soundmosaicing [9] constructs an approximation of one sound out of small pieces of other sounds using an unspecified match function without paying attention to concatenation quality.

Musical Mosaicing [10] performs a kind of automated remix of songs. It is aimed at a sound database of pop music and uses only few descriptors, but includes a measure of concatenation quality based on descriptor continuity. It uses a constraint solving approach for the selection of the units.

Concatenative singing voice synthesis based on unit selection [11, 12] relies heavily on the fixed phoneme classes and well known descriptor structure from speech synthesis. Other singing voice synthesis systems (ab)use speech synthesis techniques directly.[1]

La Légende des siècles is a theatre piece performed at the *Comédie Française* 2002, using real-time effects on the voice. One of these transformations uses a data-driven synthesis method inspired by our work: Prerecorded audio is analysed for energy and pitch as descriptors. The FFT frames are stored in a dictionary and organised into clusters, indexed by their mean descriptor values. During the performance, this dictionary is used with an inverse FFT and overlap-add to resynthesize sound with target pitch and energy given by the live audio input.

Even standard sound synthesis techniques like sampling and granular synthesis bear some characteristics of data-driven synthesis:

Sampling Modern samplers use huge amounts of sound data. Gigasampler, for instance, pride themselves to have sampled every note of a piano in every possible nuance, resulting in 1 GB of sound data. This makes samplers clearly data-driven fixed-inventory synthesis systems, with the sound database analysed by instrument class, playing style, pitch, and dynamics. The selection algorithm, however, is reduced to a fixed mapping of Midi-note

[1] `http://www.melissapop.com`

and velocity to a sample, without paying attention to the context of the notes played before, and their concatenation.

Granular synthesis is rudimentarily data-driven, but there is no analysis, the units are of constant size, and the selection is limited to choosing the position in one sound file. However, its concept of exploring a sound interactively can be combined with a pre-analysis of the data and thus enriched by a targeted selection and the resulting control over the output sound characteristics, as described in the free synthesis application in section 4.

3. THE CATERPILLAR SYSTEM

The components of the CATERPILLAR sound synthesis system are shown in figure 1 and described in the following.

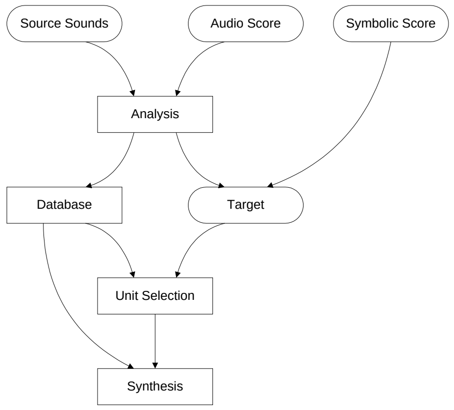

Figure 1: Overall structure of CATERPILLAR system, arrows representing flow of data.

Analysis
: The source sound files are segmented into units (3.1) and analysed for their sound and others descriptors (3.2).

Database
: Sound and data file references, descriptor definitions, units and their data are stored in a relational database (3.4).

Target
: The target specification is generated from a symbolic score (expressed in notes or descriptors), or analysed from an audio score (using the same segmentation and analysis methods as for the source sounds).

Unit Selection
: Units are selected from the database that match best the given target descriptors according to a distance function and a concatenation quality function (3.5).

Synthesis
: is done by concatenation of selected units with a short cross-fade, possibly applying transformations. Depending on the application, the selected units are placed at the times given by the target (musical or rhythmic synthesis), or are concatenated with their natural duration (free synthesis or speech synthesis).

3.1. Segmentation by Alignment

Before inclusion into the database, the source sounds have to be time-segmented into units. In the CATERPILLAR system, this is done by blind segmentation, beat segmentation, and alignment: When a Midi score is available, segmentation is done by alignment of the score with the audio signal, using a Dynamic Time-Warping algorithm, based on a peak structure distance (PSD), which matches the expected harmonic partials with the observed ones [13]. Note that this extends well to polyphonic music. Information contained in the score, e.g. note numbers or rest, the lyrics sung, or playing instructions, are attached to the units.

This way, large databases of recorded instrumental music can be automatically constituted. With the improvements for the alignment of polyphonic and multi-instrument music described in [14], a sufficient precision of 23 ms is reached.

3.2. Descriptors

A *descriptor* is a value that describes a certain quality of a sound. In speech analysis, the usual term is *feature*. In CATERPILLAR, we distinguish three classes of descriptors: *Category descriptors* are boolean and express the membership of a unit to a category or class, and all its base classes in the hierarchy (e.g. *violin* → *strings* → *instrument*). *Static descriptors* are a constant value for a unit (e.g. Midi note number), and *dynamic descriptors* are analysis data evolving over the unit (e.g. fundamental frequency).

The descriptors used in CATERPILLAR are given in the following. From the signal descriptors onward, they are defined in [15]. Of course not all of them are used in all applications.

Unit Descriptors: start and end time, duration, type

Source Descriptors: class and subclass of the sound source, playing style

Score Descriptors: Midi note pitch, polyphony, lyrics (text and phonemes), other score information or arbitrary subjective information attached to the units

Signal Descriptors: energy and logarithmic energy and their derivatives, fundamental frequency and its derivative, zero crossing rate, harmonics cutoff frequency

Perceptual Descriptors: loudness, sharpness (high frequency ratio), timbral width

Spectral Descriptors: spectral centroid, spectral tilt, spectral spread, spectral dissymmetry

Harmonic Descriptors: harmonic energy ratio, harmonic parity, tristimulus, harmonic deviation

3.3. Characteristic Values

To describe the temporal evolution of the dynamic descriptors over a unit, we reduce it to a vector of *characteristic values*:

- arithmetic mean μ and geometric mean,[2] standard deviation, minimum and maximum value, range
- slope, giving the rough direction of the descriptor movement, and curvature,[3] residual of the approximation

[2]For fundamental frequency, the geometric mean corresponds better to the perceived pitch than the arithmetic mean [16].

[3]These values are calculated by 2nd order polynomial approximation with Legendre polynomials, which have the desirable property that the lower-order polynomials are valid, albeit more coarse, approximations of the curve.

- start and end values (averaged over short windows at the beginning and end of the unit), and curve slopes at the start and end of the unit (to calculate continuity for concatenation)
- envelope: AR and inverse AR (attack and release time relative to the maximum/minimum value, respectively), ADSR envelope
- temporal center of gravity/antigravity (giving the location of the most important "elevation" or "depression" in the descriptor curve) given by $t_{\mathit{gravity}} = \sum_{i=1}^{n} t_i x_i / \sum_{i=1}^{n} x_i$, $t_{\mathit{antigravity}} = \sum_{i=1}^{n} t_i(2\mu - x_i)/\sum_{i=1}^{n} 2\mu - x_i$ for n descriptor samples, first to 4thorder temporal moments
- normalised Fourier spectrum of the descriptor in 5 bands, and the first 4 order moments of the spectrum. This reveals if the descriptor has rapid or slow movement, or if it oscillates.

3.4. Database

As the quality of the synthesis grows with the size of the sound database, an efficient database architecture is required. This is provided by using a relational DBMS (database management system). Although this results in a performance penalty for data access, the advantages in data handling prevail:

Data Independence Using a relational database, only the logical relations in the *database schema* are specified, not the physical organisation of the data. It is accessed using the declarative query language *SQL*, specifying *what* to do, not *how* to do it, leading to unprecedented flexibility, scalability and openness to change.

Consistency is assured by atomic *transactions*. A transaction is a group of SQL statements which are either completely executed, or rolled back to the previous state of the database, if an error occurred. This means no intermediate inconsistent state of the database is ever visible. Another safeguard are the consistency checks according to the relations between database tables given by the schema, and the automatic reestablishment of referential integrity by cascading deletes (for example, deleting a sound file deletes all the units referring to it, and all their data). This is also an enormous advantage while developing, because programming errors can't corrupt the database.

Client–Server Architecture Concurrent multi-user access over the network, locking, authentication, and fine-grained control over user's access permission are standard features of DBMS.

The database is clearly separated from the rest of the system by a *database interface* (see figure 2), written in *Matlab* and procedural SQL. Therefore, the DBMS used can be replaced by a different system, or other existing sound databases can be accessed, e.g. using the MPEG-7 indexing standard, or the results from the CUIDADO [17] or ECRINS [15] projects on sound content description. The database can be browsed with a graphical database explorer (*dbx*) that allows users to visualize and play the units in the database. For low-level access, we wrote C-extensions for *Matlab* that send a query to the database and return the result in a matrix. For convenience, the database interface centralises also access to external sound and data files. For the latter, the Sound Description Interchange Format (SDIF) [18] is used for well-defined exchange of data with external programs (analysis, segmentation, synthesis). The libre open source relational DBMS POSTGRESQL, finally, is reliably storing hundreds of sound and data files, tens of thousands of units, their interrelationships and descriptor data.

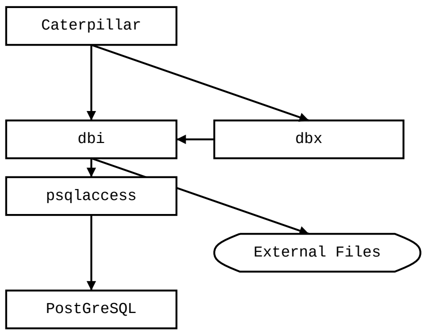

Figure 2: Architecture of the database interface (*dbi*).

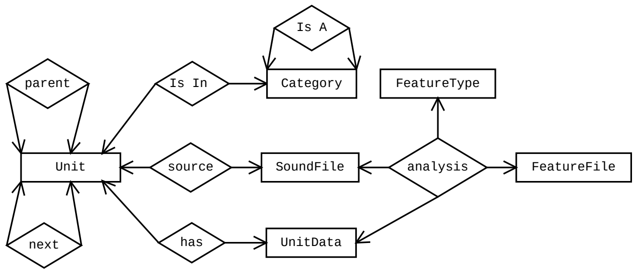

Figure 3: CATERPILLAR database schema showing entities (rectangles) and relationships (lozenges)

A simplified database schema in Entity–Relationship notation is given in figure 3. The system is open to handle all possible descriptors which can be added or changed by the user. Descriptors are either `categories` or `featuretypes` for analysed dynamic/static descriptors. Membership of a unit in a `category` is expressed as a binary descriptor by the relationship `is in`. Because the database models a containment hierarchy `parent` of units, it is enough to add the highest parent unit (eventually the whole file which is represented as a unit, too) to a class. Because we also model an inheritance hierarchy `is a` among classes, this adds all the contained units to the categories and all its base categories. For example, we have a violin recording and have added the unit that encompasses the whole file to the category *violin*. All the note units are children of the file unit. When we determine what categories a unit is in, we query its categories as given by the `is in` relationship, and those of all parents. To the obtained categories, we add all the base categories of the latter, given by the `is a` relationship.[4] The result is that all units of the violin recording are also members of the category *strings* and *instrument*, because these are the base categories of *violin*, and all this by supplying just one explicit membership.

A category can also be a *corpus*, serving to group the soundfiles or units which one desires to use for synthesis. Examples of corpora are: all violin units, pieces by Bach, pieces by Bach

[4]Modeling of tree structures in a relational database is not straightforward. In CATERPILLAR, we keep the transitive closure of the hierarchy, i.e. we store the link of a category with its direct base class, and the links to all further base classes. This also results in faster access, because all base categories can be queried immediately.

played by a certain musician, the contents of a sampling CD. We also store the name and the parameters, attached to `analysis`, of the analysis program used to perform an analysis of one or more `featuretypes` on a `soundfile`, yielding one or more `featurefiles`. For future development, the system can call the analysis programs automatically, according to stored dependencies, like the Unix *make* utility. It is convenient to store also the synthesis target in the database. For this, *virtual* soundfiles have been introduced that don't reference any external files, but serve as a container for the target with its units and their data.

3.5. Unit Selection

The classical unit selection algorithm finds the sequence of database units u^i that best match the given synthesis target units t^τ using two cost functions: The *target cost* expresses the similarity of u^i to t^τ including a context of r units around the target. The *concatenation cost* predicts the quality of the concatenation of u^i with a preceding unit u'. The optimal sequence of units is found by a Viterbi algorithm finding the best path through the network of database units.

In the following, we describe the details of the path search unit selection algorithm first proposed in [2]. The algorithm finds the units from the database of N units u_i that best match the T given synthesis target units t_τ. The quality of the match is determined by the two costs from above, given by two distance functions:

The *target cost* C^t corresponds to the perceptual similarity of the database unit u_i to the target unit t_τ. It is given as a sum of p weighted individual feature distance functions C^t_k as:

$$C^t(u_i, t_\tau) = \sum_{k=1}^{p} w^t_k \, C^t_k(u_i, t_\tau) \qquad (1)$$

To favour the selection of units out of the same context in the database as in the target, the *context cost* C^x or *extended target cost*, for the sake of the mnemonic, considers a sliding context in a range of r units around the current unit with weights w_j decreasing with distance j.

$$C^x(u_i, t_\tau) = \sum_{j=-r}^{r} w^x_j \, C^t(u^{i+j}, t^{\tau+j}) \qquad (2)$$

For most descriptors, a Euclidean distance normalised by the standard deviation is used. Some descriptors need specialised distance functions, e.g. duration is asymmetric because a longer unit can always be cut, but too short database unit can not match a longer target unit. Symbolic descriptors, e.g. phoneme class, require a lookup table of distances.

The *concatenation cost* C^c expresses the discontinuity introduced by concatenating the units u_i and u_j from the database. It is given by a weighted sum of q feature concatenation cost functions C^c_k:

$$C^c(u_i, u_j) = \sum_{k=1}^{q} w^c_k \, C^c_k\,(u_i, u_j) \qquad (3)$$

The cost depends on the unit type: concatenating an attack unit allows discontinuities in pitch and energy, a sustain unit does not. Consecutive units in the database (the pairs that are in relationship `next` in figure 3) have a concatenation cost of zero. Thus, if a whole phrase matching the target is present in the database, it will be selected in its entirety, leading to nonuniform unit selection.

We use, for instance, the distance between pitch end value of the left and pitch start value of right unit plus the difference in slope, for the concatenation of two units in their sustain phase. For pitch, we can take the descriptor's FFT spectrum into account to match similar vibrato frequency and intensity.

The unit database can be seen as a fully connected state transition network through which the unit selection algorithm has to find the least costly path that constitutes the target. Using the weighted extended target cost $w^t C^x$ as the *state occupancy cost* b_{ij}, and the weighted concatenation cost $w^c C^c$ as the *transition cost* a_{ij}, the optimal path can be efficiently found by a Viterbi algorithm.

```
for $1 \leq j \leq N$:
    $a_{j1} = C^x(t_1, u_j)$
for $2 \leq \tau \leq T$:
    for $1 \leq j \leq N$:
        $b_{j\tau} = w^t C^x\,(t_\tau, u_j)$
        $a_{j\tau} = \min_{1 \leq k \leq N} (a_{k,\tau-1} + w^c C^c\,(u_k, u_j) + b_{j\tau})$
        $\psi_{j\tau} = k_{\min}$
```

The decoding of the path ψ to find the optimal sequence of units s_τ is done in reverse order by first finding the path endpoint k with the least global cost a_{kT}, and then following the backward indices:

```
$k = \arg\min_{1 \leq j \leq N}(a_{jT})$
for $T \geq \tau \geq 1$:
    $s_\tau = u_k$
    $k \ = \psi_{\tau k}$
```

3.6. Unit Selection by Constraint Satisfaction

Although the path-search unit selection algorithm shows good results, the algorithm is too rigid and it is hard to integrate other requirements in a flexible way, e.g. replacing just one displeasing unit in a sequence proposed by CATERPILLAR, or forcing all selected units to be different. That's why we reformulated the selection algorithm as a constraint satisfaction problem (CSP) using the *adaptive local search* (ALS) algorithm [19, 20]. Constraint satisfaction is defined by an error function, which allows us to easily express the unit selection algorithm as a CSP using the target and concatenation costs between units. Indeed, one can argue that path-search unit selection is a special case of adaptive local search unit selection where each target unit is visited only once.

A problem in CSP form is given by:

The *variables* V_i, for which we seek a configuration that satisfies our constraints, are the sequence of unit indices to select. The *global cost function* to minimise is the total selection cost:

$$\begin{aligned} C^s = {} & w^t C^x\,(u_{V_1}, t_1) + \\ & \sum_{i=2}^{T} w^c C^c\left(u_{V_{i-1}}, u_{V_i}\right) + w^t C^x\,(u_{V_i}, t_i) \end{aligned} \qquad (4)$$

Each *unit constraint* $\mathbf{C}_i$ comprises the variables V_{i-1} through V_{i+1}. The error function for one constraint is given by the target cost for the unit u_{V_i}, and the concatenation cost to the adjacent units $u_{V_{i-1}}$ and $u_{V_{i+1}}$ in the current configuration.

$$\begin{aligned} E(\mathbf{C}_i) = {} & w^t C^x\,(u_{V_i}, t_i) + \\ & w^c C^c\left(u_{V_{i-1}}, u_{V_i}\right) + w^c C^c\left(u_{V_i}, u_{V_{i+1}}\right) \end{aligned} \qquad (5)$$

The adaptive local search algorithm starts from a random configuration of variables and repeats the following steps until the global cost drops under a given threshold, or a maximal number of iterations is reached:

1. For each constraint, compute its error and distribute it over the variables appearing on the constraint. A variable V_i is responsible for its target error $w^t C^x\left(u_{V_i}, t_i\right)$ and half of each concatenation error $w^c C^c\left(u_{V_{i-1}}, u_{V_i}\right)$, $w^c C^c\left(u_{V_i}, u_{V_{i+1}}\right)$, which are summed.
2. The unit in the variable with the highest error not marked as taboo is replaced by the best matching unit u_j with

$$j = \arg\min_{1\leq j\leq N}\left(\begin{array}{l} w^t C^x\left(u_{V_i}, t_i\right) + \\ w^c C^c\left(u_{V_{i-1}}, u_{V_i}\right) + \\ w^c C^c\left(u_{V_i}, u_{V_{i+1}}\right) \end{array}\right) \tag{6}$$

3. If no unit with lower error exists, mark the variable as taboo for a given number of iterations.
4. If all variables are marked taboo, restart with a new random configuration.

Additional constraints are:

- The *all different* constraint says that no unit must appear twice in the selection. It distributes a high penalty over the variables containing repeated units.
- *Unit ban* adds a high penalty to the variables containing a banned unit.
- *Unit forcing* lowers the target error of variables containing the units to include at the desired position.
- *Unit lock* marks the variable with the unit to lock as taboo.

To summarise, with CSP unit selection we sacrifice the real-time synthesis oriented efficient[5] linear selection of units in the path-search unit selection for a more composer-oriented interactive choice approach. This adds computational complexity, but also adds flexibility and interactivity.

4. APPLICATIONS

High Level Instrument Synthesis Because the CATERPILLAR system is aware of the context of the database as well as the target units, it can synthesise natural sounding transitions by selecting units from matching contexts. Information attributed to the source sounds can be exploited for unit selection, which allows high-level control of synthesis, where the fine details lacking in the target specification are filled in by the units in the database.

We use a database of seminotes, i.e. each note segment found by alignment with the score is split in half and added as two units to the database. Ideally, the split point should be determined by the zone of greatest stability. The end seminote from a note and the beginning seminote of the following note form a transition unit (the term *dinote* as in *diphone* seems too awkward). Transition units are the basic units of selection and can be more easily concatenated because the sound is more stable at their edges.

We use a database made from pieces for solo violin (J.S. Bach's *Sonata and Partita*, over one hour of music, played by different violinists).

[5]The asymptotical complexity of the dynamic programming path-search unit selection algorithm is $O(TN^2)$.

Free synthesis from heterogeneous sound databases offers a sound composer efficient control of the result by using perceptually meaningful descriptors. This type of synthesis is interactive and iterative. The CATERPILLAR system supports this by its graphical database browser and the ability—by adding the appropriate constraints—to freeze good parts from a synthesis and regenerate others, or to force specific units to appear in the synthesis.

Resynthesis of audio with sounds from the database: A sound or phrase is taken as the audio score, which is resynthesized with the same pitch, amplitude, and timbre characteristics using units from the database.

Loop Based Synthesis In electronic dance music, a large part of the musical material comes from sampling CDs, containing rhythmic loops and short bass or melodic phrases. As the published CDs number in the thousands, each containing hundreds of samples, a large part of the work consists in listening to the CDs and select suitable material. A database would come handy, that stores loops by given categories (rhythmic style and bpm information is usually supplied by the CD description), and automatically analysed acoustic and perceptive descriptors (high frequency content, sharpness, roughness, dissonance, etc.). The selection algorithm would then supply a sequence of loops satisfying a given dynamical evolution, the concatenation distance being controlled by the desire for continuity or change in the piece.

Artistic Speech Synthesis An interesting current project uses CATERPILLAR to recreate the voice of a defunct eminent writer to read one of his texts for which no recordings exist. The goal here is different from fully automatic text-to-speech synthesis: highest speech quality is needed (concerning both sound and expressiveness), manual refinement is allowed. The role of CATERPILLAR is to give the highest possible automatic support for human decisions and synthesis control, and to select a number of well matching units in a very large base (obtained by automatic alignment) according to high level linguistic descriptors, which reliably predict the low-level acoustic characteristics of the speech units from their grammatical and prosodic context [6], and emotional and expressive descriptors.

The adaptations that had to be applied to the CATERPILLAR system to reach this aim were easy to integrate, proving the flexibility of the system and the data organisation:

- Addition of linguistic descriptors, calculated by the text analysis part of the EULER TTS system [5]
  - phonetic descriptors (phoneme class and super-classes)
  - phonological descriptors (containing syllable, word, sentence, and positions of the unit in these)
  - syntactical descriptors (grammatical nature and function of containing word)
- Integration of new target and concatenation distance functions for the new descriptors into the unit selection algorithm. The basic unit is a *semiphone*, two of which are joined into a diphone, because diphone concatenation yields better quality. However, if a needed diphone is very rare, or no diphone in an appropriate context can be found, the selection falls back to semiphones.
- Exploitation of the phoneme classes added to the database for faster unit search. This is is handled by the DBMS automatically by the use of indices.

5. CONCLUSIONS AND FUTURE WORK

The CATERPILLAR system shows good results in instrument synthesis, and surprising sounds from free synthesis, and will soon be tested with artistic speech synthesis. The concept of high-level musical synthesis is confirmed by these results. More evidence for text-to-speech synthesis is given in [6].

Professional and multi-media sound synthesis devices or software show a natural drive to make use of the advanced mass storage capacities available today. We can foresee this type of applications hitting a natural limit of manageability of the amount of data. Only automatic support of the data-driven composition process will be able to surpass this limit and make the whole wealth of musical material accessible to the musician.

In CATERPILLAR, the integration of an SQL database was a quantum leap forward for the ease of data handling. Finally, the use of a constraint satisfaction framework allows to combine the requirements for creative synthesis with the proven unit selection algorithm derived from speech synthesis, keeping maximum flexibility.

Adaptive Target Re-segmentation CSP unit selection allows one more important improvement impossible with path search unit selection: the target can be automatically re-segmented when the unit selection cost C^s remains too high. When, after a certain number of iterations of the ALS unit selection algorithm, a unit's cost cannot be improved, and the cost is above a re-segmentation threshold Θ_r, we try to split the target unit into two new units, in the hope that we can find two smaller units that together form a better match for the original target unit.

This is especially well applicable to the resynthesis of audio application, since then target segmentation errors have less influence on the result because they can be corrected. We can even imagine to start with only one target unit for the whole target sound and have the unit selection algorithm develop the match and the segmentation. However, a method for an efficient choice of the split point needs to be found. Note that a sort of backtracking, i.e. undoing of a target unit split, is automatically included by favouring the selection of contiguous units from the database.

Automatic Weight Training improves the weights for the feature distance functions automatically by an exhaustive search over the weight space, maximising the acoustic similarity of an original sound used as target and the synthesis result [21]. This would come handy for the many descriptors in CATERPILLAR where manual weight tuning proved tedious sometimes.

6. REFERENCES

[1] Diemo Schwarz, "A System for Data-Driven Concatenative Sound Synthesis," in *Digital Audio Effects (DAFx)*, Verona, Italy, 2000.

[2] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proc. ICASSP*, Atlanta, GA, May 1996.

[3] Alan Black, Paul Taylor, and Richard Caley, "The Festival Speech Synthesis System: System Documentation," Technical Report HCRC/TR-83, Human Communication Research Centre, 1998.

[4] M. Beutnagel et al., "The AT&T Next-Gen TTS System," in *Joint Meeting of ASA, EAA, and DAGA*, Berlin, 1999.

[5] Michel Bagein et al., "Le projet EULER, Vers une synthèse de parole générique et multilingue," *Traitement automatique des langues*, vol. 42, no. 1, 2001.

[6] Romain Prudon and Christophe d'Alessandro, "A selection/concatenation TTS synthesis system: Databases developement, system design, comparative evaluation," in *4th Speech Synthesis Workshop*, Pitlochry, Schotland, 2001.

[7] John Oswald, "Plunderphonics," web page, 1999, `http://www.plunderphonics.com`.

[8] Reynald Hoskinson and Dinesh Pai, "Manipulation and resynthesis with natural grains," in *Proceedings of the ICMC*, Havana, Cuba, 2001.

[9] Steven Hazel, "Soundmosaic," web page, 2001, `http://thalassocracy.org/soundmosaic`.

[10] Aymeric Zils and François Pachet, "Musical Mosaicing," in *Digital Audio Effects (DAFx)*, Limerick, Ireland, Dec. 2001.

[11] M. W. Macon et al., "Concatenation-Based MIDI-to-Singing Voice Synthesis," in *103rd Meeting of the AES*. 1997, New York.

[12] Yoram Meron, *High Quality Singing Synthesis Using the Selection-based Synthesis Scheme*, Ph.D. thesis, University of Tokyo, 1999.

[13] Nicola Orio and Diemo Schwarz, "Alignment of Monophonic and Polyphonic Music to a Score," in *Proceedings of the ICMC*, Havana, Cuba, 2001.

[14] Ferréol Soulez, Xavier Rodet, and Diemo Schwarz, "Improving Polyphonic and Poly-Instrumental Music to Score Alignment," in *ISMIR*, 2003.

[15] Xavier Rodet and Patrice Tisserand, "ECRINS: Calcul des descripteurs bas niveaux," Tech. Rep., Ircam – Centre Pompidou, Paris, Oct. 2001.

[16] J. I. Shonle and K. E. Horan, "The pitch of vibrato tones," *JASA*, vol. 67, no. 1, pp. 246–252, 1980.

[17] H. Vinet, P. Herrera, and F. Pachet, "The Cuidado Project: New Applications Based on Audio and Music Content Description," in *Proc. ICMC*, Gothenburg, 2002.

[18] D. Schwarz and M. Wright, "Extensions and Applications of the SDIF Sound Description Interchange Format," in *Proc. ICMC*, Berlin, Aug. 2000.

[19] Philippe Codognet and Daniel Diaz, "Yet another local search method for constraint solving," in *AAAI Symposium*, North Falmouth, Massachusetts, 2001.

[20] Charlotte Truchet, Gérard Assayag, and Philippe Codognet, "Visual and adaptive constraint programming in music," in *Proc. ICMC*, Havana, Cuba, 2001.

[21] Yoram Meron and Keikichi Hirose, "Efficient weight training for selection based synthesis," in *EUROSPEECH*, Budapest, Hungary, 1999.