A SYSTEM FOR DATA-DRIVEN CONCATENATIVE SOUND SYNTHESIS

Diemo Schwarz

IRCAM – Centre Pompidou
Analysis–Synthesis Team[8]
1, place Igor–Stravinsky, 75004 Paris, France
schwarz@ircam.fr

ABSTRACT

In speech synthesis, concatenative data-driven synthesis methods prevail. They use a database of recorded speech and a unit selection algorithm that selects the segments that match best the utterance to be synthesized. Transferring these ideas to musical sound synthesis allows a new method of high quality sound synthesis. Usual synthesis methods are based on a model of the sound signal. It is very difficult to build a model that would preserve the entire fine details of sound. Concatenative synthesis achieves this by using actual recordings. This data-driven approach (as opposed to a rule-based approach) takes advantage of the information contained in the many sound recordings. For example, very naturally sounding transitions can be synthesized, since unit selection is aware of the context of the database units. The CATERPILLAR software system has been developed to allow data-driven concatenative unit selection sound synthesis. It allows high-quality instrument synthesis with high level control, explorative free synthesis from arbitrary sound databases, or resynthesis of a recording with sounds from the database. It is based on the new software-engineering concept of component-oriented software, increasing flexibility and facilitating reuse.

1. INTRODUCTION

Concatenative data-driven synthesis methods use a large database of source sounds, segmented into *units*, and a *unit selection* algorithm that finds the units that match best the sound or phrase to be synthesised, called the *target*. The selection is performed according to the features of the units. These are characteristics extracted from the source sounds, e.g. pitch, or attributed to them, e.g. phoneme class. The selected units are then transformed to fully match the target specification, and 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.

Concatenative synthesis systems based on unit selection use a *data-driven* approach, as opposed to rule based approaches. Instead of supplying rules constructed by careful thinking, the rules are induced from the data itself.

After a more detailed introduction into unit selection in speech (1.1) and musical synthesis (1.2), and an attempt on a comparison of the two (1.3), the bulk of the article is dedicated to the concatenative data-driven sound synthesis system called CATERPILLAR, developed by the author (section 2), followed by a few words on software-engineering and system architecture (section 3), applications (section 4), future work (section 5), and conclusions (section 6).

1.1. Unit Selection in Speech Synthesis

Concatenative unit selection speech synthesis from large databases is used in a great number of Text-to-Speech systems for waveform generation [1, 2, 3]. Its introduction resulted in a considerable gain in quality of the synthesized speech. Unit selection algorithms attempt to predict the appropriateness of a particular database speech unit using linguistic features predicted from a given text to be synthesized. The units can be of any length (non-uniform unit selection), from sub-phonemes to whole phrases, and are not limited to diphones or triphones.

Those data-driven synthesis systems are generally considered superior to rule-based parametric synthesis systems concerning naturalness and intelligibility. Indeed, findings in other domains, for instance in speech recognition [4], corroborate the general superiority of data-driven approaches.

1.2. Unit Selection in Musical Synthesis

Despite its promising approach and its success in speech synthesis systems, concatenative data-driven methods are rarely used in musical synthesis. There are first attempts on singing voice synthesis [5], and partial applications of data-driven methods to parametric synthesis, e.g. [6].

Application of the approach of data-driven unit selection synthesis from speech synthesis to musical sound synthesis allows a new method of **high quality sound synthesis**. 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 the fine details of the sound. On the contrary, concatenative synthesis, by using actual recordings, preserves entirely the fine details of sound. If the database of recordings is large enough, a great number of sound events in many different contexts are available, so that there is rarely the need to apply transformations, which always degrade the sound. The data-driven approach takes advantage of the information contained in the many sound recordings. For example, very naturally sounding transitions can be synthesized, since unit selection is aware of the context of the database units, and selects a unit containing the transition in the desired target context.

However, musical creation is an artistic activity, and is only partly based on clearly defined criteria. Therefore, creative use of the concatenative unit selection synthesis method is supported through interactive and iterative **free synthesis** by the CATERPILLAR system. This means that composers, musicians, and multimedia creators can explore unit selection sound synthesis using perceptually meaningful control features and high-level descriptions of the source sounds. They can discover new sounds by “browsing” arbitrary sound databases (not necessarily containing musical instruments). The system integrates them easily and is extensible to additional features and new selection algorithms.

1.3. Comparison of Musical Synthesis and Speech Synthesis

Even from a very rough comparison between musical and speech synthesis, some profound differences spring to mind, which make the application of concatenative data-driven synthesis techniques from speech to music non-trivial:

- Speech is a-priori clustered into phonemes. In tonal music, there are pitch classes, but in general, no clustering can be presupposed.
- In speech, the time position of synthesized units is intrinsically given by the natural duration of the selected units. In music, precise time-points have to be hit.
- In speech synthesis, intelligibility and naturalness are of prime interest, and the synthesised speech is often limited to "normal" informative mode. Musical synthesis uses many modes of expressivity, and needs to experiment.

2. THE CATERPILLAR SYSTEM

The CATERPILLAR software system described in this section has been developed to perform data-driven concatenative unit selection sound synthesis.

Figure 1 shows a structural overview of the system. The subtasks that it performs and that will be explained in detail in the sections indicated in parenthesis, are:

- Segmentation (2.1) and analysis into features (2.2) of the source sounds, and possibly of an audio score
- Generation of the target specification from a symbolic score or from an audio score (2.3)
- Handling of the sound and data files, and of the units in the database (2.4)
- Selection of units from the database according to given target features and an acoustic distance function (2.5)
- Waveform synthesis by concatenation of selected units, possibly applying transformations (2.6)

2.1. Segmentation

Before inclusion into the database, the source sounds have to be time-segmented into units. In the CATERPILLAR system, this is done in various ways:

- By an external segmentation program described in [7], integrated as a component (see section 3), which performs note or phone segmentation.
- When a score is given with the audio, e.g. from a MIDI file, segmentation is done by alignment of the pitch contour, using a Dynamic Time-Warping algorithm. Information contained in the score, e.g. MIDI note number or rest, the lyrics sung, or playing instructions, are attached to the unit. This symbolic data can later be exploited for unit selection.
- Audio with a steady pulse is segmented into regularly spaced units, given some hit-points in time and the desired subdivision.

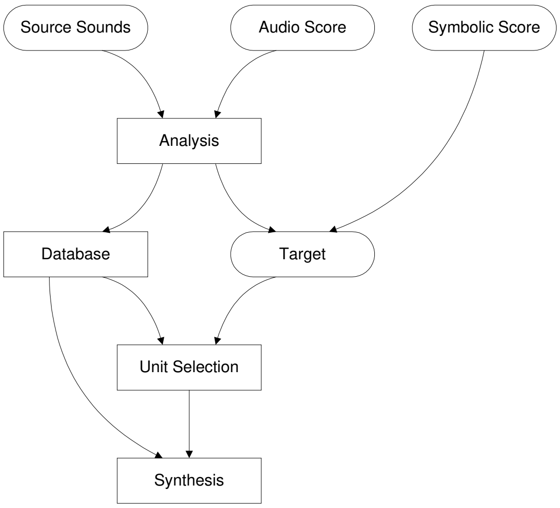

Figure 1: Overall structure of the data-driven CATERPILLAR system, arrows representing flow of data.

- At last, segmentation can of course be manual, or manually edited automatic segmentation.

On each note segment, a *sub-segmentation* is performed that further subdivides it into an attack phase, followed by a sustain phase, and a release phase [8]. The release phase and the attack phase of the following unit together form a transition segment. For the moment, the attack phase is blindly set to 40 ms into the segment, the release phase 40 ms before the end of the segment.

2.2. Analysis and Features

The source sound files are analysed by standard signal processing methods. The analysis data for each unit is then used to calculate the scalar features that describe it. Analysis is performed mostly by external programs called as components (see section 3). They perform, amongst others, analysis of pitch, energy, spectrum, additive partials, and spectral envelope [9].

Features are characteristics analysed from, or attributed to sound signals. There are three classes of features: continuous, discrete, and symbolic:

Continuous features are calculated from the analysis data over the duration of one unit. The raw data is used to compute a vector of *characteristic values* for that unit and that feature, containing:

- the mean feature value
- the standard deviation
- minimum, maximum, and range of the feature
- the average slope, giving the rough direction of the feature movement
- the normalised frequency spectrum of the feature in bands, and the first 4 order moments of the spectrum. This reveals if the feature has rapid or slow movement, or if it oscillates (see figure 2).

The continuous features used for selection are: pitch, energy, spectral tilt, spectral centroid, spectral flux, inharmonicity, and voicing coefficient. How these are computed is explained in [7].

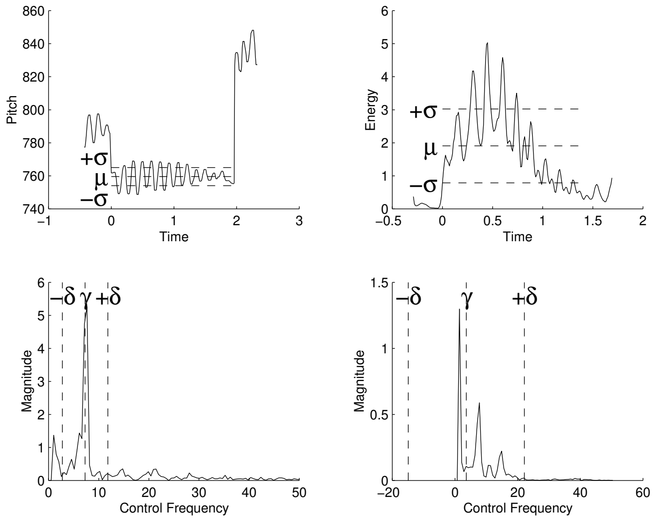

Figure 2: Example of characteristic values of continuous features: The raw features (pitch and energy) result in mean μ and standard deviation σ over the duration of the unit, indicated by the length of the dotted lines (top), and magnitude spectrum of the feature, spectral centroid γ, and second order moment δ (bottom).

Discrete features take one numeric value per unit, e.g. MIDI note number or number of notes played. Information about the segment the unit came from, namely sound file id, start time in the sound file, and unit duration, are available as discrete features to the selection algorithm. This serves to address a specific unit, which is useful for free synthesis when, at a certain target time, a composer wants to force a specific unit to appear, or restrict the selection algorithm to one source sound file.

Symbolic features take one symbolic value per unit, given by or derived from the symbolic score. They are implemented as discrete features with integer values.

The symbolic features used in CATERPILLAR are the phoneme sung for a voice signal, the instrument class and subclass, the notes played, and expressive performance instructions, such as staccato, legato, trill, and dynamics (crescendo, diminuendo). A special symbolic feature is the *unit type*, which can take the values *pause* for a rest, *note* for a complete note unit, *attack*, *sustain*, or *release* for sub-segmented units, or *phrase* for multi-note units. This way, the unit type is available for selection.

Feature extensibility. The system is open to handle any possible feature. It is extensible, so that features can be added or changed dynamically. See section 3 for more details. Additionally, any feature can be manually attributed to the units, be it subjective features (say, "glassiness"), or other information not automatically derivable from the sounds.

2.3. Target Specification

The target features can come from two different sources, or be a combination from both:

A **symbolic score**, e.g. from a MIDI file, contains discrete note values and other control parameters, such as volume or brilliance. Also, high-level performance instructions, e.g. legato or staccato, and the lyrics sung for a voice piece can be attached to the symbolic score.

An **audio score** is a recording that is analysed to obtain certain target features. Taking the complete target features from the audio score allows resynthesizing it with sounds from the database.

To be usable by the unit selection algorithm, the target specification has to be segmented into a sequence of target units t^τ and the target unit features t^τ_f have to be generated. A symbolic score is segmented by the notes it contains, and the target features are generated from the symbolic information. An audio score is segmented and analysed just like sounds for the database (see section 2.1). The resulting sequence of target units in either case is sub-segmented into attack, sustain, and release phase, as for database units, to give the unit selection algorithm more freedom in combining transitions and sustained parts.

2.4. Database

The database holds references to the original sound files and to the data files from analysis. It stores the units and the unit features.

As test case for high quality instrument synthesis, a database made from pieces for solo violin (J.S. Bach's *Sonata and Partita*, over one hour of music, played by different violinists) is used. For creative synthesis, various recordings of environmental, instrumental, voice, and electronic sounds are used.

A prototypical database subsystem has been implemented on flat analysis data files and SDIF [10] unit feature files. The database is clearly separated from the rest of the system, and solely accessed via a *database driver* interface. Therefore, the simple database can later be replaced by a full relational DBMS (database management system), or use other existing sound databases. For instance, sound databases using the emerging ISO MPEG-7 standard [11, 12] for indexing could be used.

The database can be browsed with the database explorer (figure 3), written in *Matlab*, that allows users to visualize and play the units in the database.

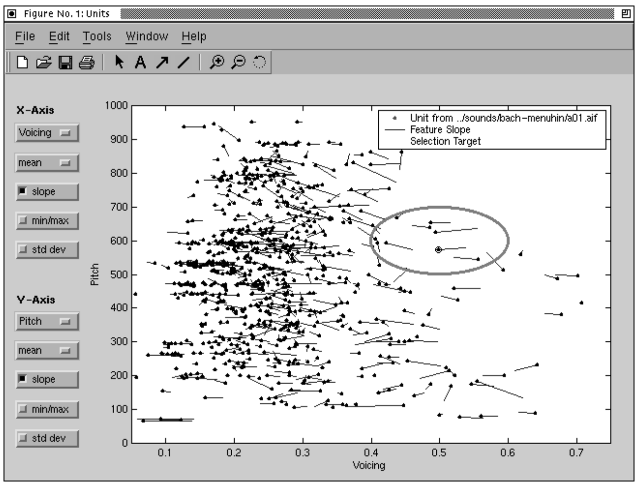

Figure 3: Database explorer feature view: Each point represents a unit, plotted according to two selectable characteristic values of two features. Various characteristic values can be displayed with the units, e.g. min/max, the standard deviation, or the mean slope (the short lines extending from the units). The ellipse serves to interactively select the units for real-time acoustic exploration of the database. The currently played unit within the ellipse is highlighted by a little circle.

2.5. Unit Selection

The starting point of the unit selection algorithm is a database of units u^i and a sequence of target units t^τ, both given by n unit feature values[1] u^i_f and t^τ_f. The unit selection algorithm incrementally finds the units from the database that best match the given synthesis target units. The quality of the match is determined by two values, expressed as costs:

The **target cost** p^i_c corresponds to the perceptual similarity of the database unit i to the current target unit τ. It is given by a perceptual distance function D as:

$$p^i_t = \sum_{j=-r}^{r} w_j\, D(u^{i+j}, t^{\tau+j}, j) \qquad (1)$$

where the "sliding" context in a range of r units around the current target unit t^τ is compared with the context around the database unit u^i, with weights w_j decreasing with distance j (see figure 4).

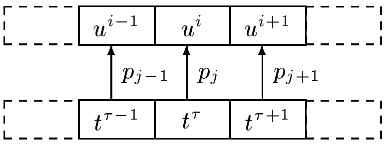

Figure 4: Target context sub-costs $p_j = D(u^{i+j}, t^{\tau+j}, j)$ of p^i_t for a context range of $r = 1$.

D is a weighted sum of individual feature distance functions d_f:

$$D(u, t, j) = \sum_{f=1}^{n} w_f\, d_f(u, t, j) \qquad (2)$$

For discrete and symbolic features, a feature distance matrix is used. The feature distance functions may be non-symmetric, e.g. when comparing the feature *unit duration*, a database unit longer than the target unit has a feature distance of zero, because it can be shortened.

Note that each feature distance function d_f has access to all features of the unit to take care of interdependencies between features. It is also aware of the context position, because the influence of the context may depend on the values of certain features, e.g. unit type.

The target cost also expresses the distortion introduced by possibly necessary transformation of the database unit to fully match the target unit.

The **concatenation cost** p^i_c predicts the discontinuity introduced by concatenation of the unit i from the database with the preceding selected unit u' (see figure 5). It is given by a concatenation cost function C, which is a weighted sum of feature concatenation cost functions c_f:

$$p^i_c = C(u', u^i) = \sum_{f=1}^{n} w_f\, c_f(u', u^i) \qquad (3)$$

The concatenation 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 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 non-uniform unit selection. This also favours an expansion of a sub-unit to its following sub-unit.

Combining the costs. Costs are combined in a weighted sum of sub-costs to form the selection cost p^i_s for unit i:

$$\begin{aligned} p^i_s &= w_c\, p^i_c + w_t\, p^i_t \\ &= w_c\, C(u', u^i) + w_t \sum_{j=-r}^{r} w_j\, D(u^{i+j}, t^{\tau+j}, j) \qquad (4) \\ &= w_c \sum_{f=1}^{n} w_f\, c_f(u', u^i) + w_t \sum_{j=-r}^{r} w_j \sum_{f=1}^{n} w_f\, d_f(u^{i+j}, t^{\tau+j}, j) \end{aligned}$$

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 [2]. Using the weighted target cost $w_t p^i_t$ as the *state occupancy cost*, and the weighted concatenation cost $w_c p^i_c$ as the *transition cost*, the optimal path can be efficiently found by a Viterbi algorithm (see figure 5).

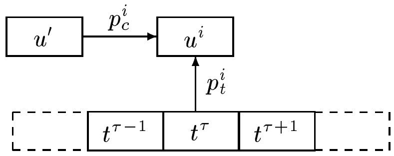

Figure 5: Concatenation cost p^i_c and target cost p^i_t in unit selection.

The CATERPILLAR system currently uses a weighted Euclidean distance function on the feature values, normalized by division by the standard deviation.

Clustering. Finding the unit closest to a target unit, performing an exhaustive search is too slow when the database is large. As an alternative, a k-nearest-neighbours algorithm is used, i.e. the search space is split up into a tree of hypercubes. To further improve the performance of unit selection for large databases, the data can be clustered, e.g. by the CART method (Classification and Regression Trees, [13]), as used in speech synthesis [1, 14, 15]. Also, the search space can be pruned at each step: only the n closest units according to the target cost p^i_t are included in the path search.

2.6. Synthesis

The synthesis is performed in two steps: transformation and concatenation:

Transformation is done, if necessary, by one or more transformation components to fully match the target features. The need and the extent of transforming the units is reduced with a large database. Therefore, simple methods are sufficient. Transformation can affect the fundamental frequency, the energy, or the spectral envelope. Pitch is transformed by resampling, energy transformation by multiplication, and the spectral envelope is changed by filtering to approach the given spectral tilt or centroid.

For **concatenation**, a simple method is used for the moment, implemented in a concatenation component: The units' audio segments are joined with a slight overlap, and a crossfade of the overlapping part is performed.

[1] For the sake of simplicity, the different characteristic values of a continuous feature are considered as individual features here.

3. SOFTWARE ARCHITECTURE

The application of software-engineering principles bears multiple advantages also for research. They are the key to the successful development of a complex software system like CATERPILLAR. The software-architecture is built according to an object-oriented design with clearly separated subsystems, and relies on re-usable components for improved flexibility and maintainability.

In the new concept of *component-oriented software* [16], a system is built out of interacting components. Components are independent binary units, whose interface is formally and contractually defined. They are expected to be the cornerstone of software in the years to come. Indeed, looking at the software in use today, one can see that the successful programs are often those that are open to components, e.g. as plugins.

The most important advantage of components is that they facilitate reuse of software-parts, thus reducing development cost, effort, and time. They avoid some problems of other methods of reuse, such as function or class libraries. The binary deployment assures a high level of encapsulation, allowing better separation of concerns, and fewer problems when changing components. Very modular and flexible systems are possible through *late binding*, i.e. dynamic loading of one of several similar components chosen at run-time. This allows easy comparison of different algorithms.

In the CATERPILLAR system, components are used for transparent analysis-on-demand: An analysis component, which reads sound descriptions and output features, is only started when a feature it computes is actually accessed. The results are stored persistently in the database. As the control parameters for analysis are also in the database, adding sounds and features, and changing components and parameters becomes completely transparent. This means, that the user does not have to restart the analysis of the whole data manually if a parameter or a component was changed (with the risk of forgetting to re-analyse some files and leaving inconsistent data around).

As the component interface also includes the dependencies on the output of other components (e.g. additive harmonic analysis needs spectral peaks and fundamental frequency), the required data will also be recalculated, when needed. What's more, because the component interface is formally defined, user interfaces (command-line or graphical) for parameter input can be automatically generated.

With regard to an easy applicability to the multi-media market, standardized open interfaces and exchange formats have been developed and used in the CATERPILLAR system:

- The document standard XML is used for configuration files, database description, and definition of the component interfaces.
- MIDI is used for the scores, so that standard music software can be used to edit target specifications.
- The portable and extensible SDIF *Sound Description Interchange Format* [10] is used for importing and exporting sound data and features, for data interchange between the components, and to store segments and units.

4. APPLICATIONS

The practical applications of CATERPILLAR include:

High quality instrument synthesis from MIDI. Because the CATERPILLAR system is aware of the context of the database as well as the target units, it can create much more natural sounding transitions than a sampler or any non-physical-modeling synthesizer. 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.

Free synthesis from arbitrary sound databases offers a sound composer efficient control of the result by using perceptually meaningful features, such as spectral centroid. This type of synthesis is interactive and iterative. The CATERPILLAR system supports this by its graphical database browser and the ability 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 and annotated with additional information such as instrument type. It is then resynthesized with the same pitch, amplitude, and timbre characteristics using units from the database. One could, for example automatically replace drum sounds in a recording.

Moreover, the central techniques of selection from large sound databases, that are implemented in the CATERPILLAR system, can be fruitfully used for other applications as well: For content based retrieval of certain parts of sound recordings, or for research of characteristics of musical expression (*gestures*).

5. FUTURE WORK

Future work includes fine-tuning of the distance function. Obviously, there are far too many weights to be sensibly tuned by hand[2]. Instead, training the weights and thus the cost functions from the database has to be considered, as described in [1, 2].

More features will be added, such as perceptually motivated features related to a timbre space, phase coupling [17], and features from bispectral higher order statistics [18].

To derive realistic target features from a symbolic score for a given instrument, performance rules [19] could be studied. Alternatively, the symbolic score could be converted to an audio score by playing it with a sample-based synthesizer. This would deliver quite realistic target features for energy, spectral envelope, etc.

Alternatively, more information-rich score formats have to be examined, for instance the *Standard Music Description Language* SMDL [20], or others, summarized in [21]. However, there is a much smaller number of works available than in MIDI file format.

The hypothesis that sub-segmentation (the subdivision of a unit into many smaller units) doesn't negatively affect the quality of the synthesis must be verified.

The quality of the concatenation might be improved by optimizing the join points, either during synthesis, as suggested in [22], or by pre-analysing the database as described in [23].

The need for integration of a PSOLA transformation component for synthesis has to be studied [24]. It would allow independent pitch manipulations and time-stretching with little degradation of the signal.

Concatenation using parametric interpolation on an additive sinusoids plus residual signal model (harmonics plus noise) has to be considered. This would interpolate the sinusoidal partials and the residual spectral envelope. See [25] for an application to concatenative speech synthesis. Transformations would then become straightforward.

[2]What's more, that amounts to falling back to a rule-based approach.

6. CONCLUSIONS

The CATERPILLAR system described in this article shows very promising results in instrument synthesis, and surprising sounds from free synthesis.

As is to be expected from the early applications of unit selection speech synthesis, the result is good most of the time, but there are some very bad cases, where inappropriate units have been selected. However, for musical composition, this doesn't matter so much, because certain units can interactively be forced to appear or not to appear.

The CATERPILLAR system proves that applying the concept of data-driven concatenative synthesis based on non-uniform unit selection to musical synthesis is a valid approach.

7. ACKNOWLEDGEMENTS

The author wishes to thank Xavier Rodet, Ross Ballany, and Guillaume Lemaitre for their valuable support in writing this article.

8. REFERENCES

[1] M. W. Macon, A. E. Cronk, and J. Wouters, "Generalization and discrimination in tree-structured unit selection," in *Proceedings of the 3rd ESCA/COCOSDA International Speech Synthesis Workshop*, November 1998, www[3].

[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, www[4].

[3] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal, "The AT&T Next-Gen TTS System," in *Joint Meeting of ASA, EAA, and DAGA*, Berlin, Germany, Mar. 1999, www[5].

[4] H. Hermansky, "Data-Driven Speech Analysis for ASR," in *Proceedings of the First Workshop on Text, Speech, Dialogue TSD'98*, P. Sojka et al., Eds., Brno, Czech Republic, Sept. 1998, Masaryk University Press, www[6].

[5] M. W. Macon et al., "Concatenation-Based MIDI-to-Singing Voice Synthesis," in *103rd Meeting of the AES*. 1997, New York, www[3].

[6] D. Wessel, C. Drame, and M. Wright, "Removing the Time Axis from Spectral Model Analysis-Based Additive Synthesis: Neural Networks vs. Memory-Based Machine Learning," in *Proc. ICMC*, Ann Arbor, Michigan, 1998, www[7].

[7] S. Rossignol, *Segmentation et indexation des signaux sonores musicaux*, Ph.D. thesis, Univ. Paris VI, July 2000, www[8].

[8] J. Hajda, "A New Model for Segmenting the Envelope of Musical Signals: The relative Salience of Steady State versus Attack, Revisited," in *Journal of the AES*, Nov. 1996.

[9] D. Schwarz and X. Rodet, "Spectral Envelope Estimation and Representation for Sound Analysis-Synthesis.," in *Proc. ICMC*, Beijing, Oct. 1999, www[8].

[3] `http://cslu.cse.ogi.edu/tts`
[4] `http://www.cstr.ed.ac.uk`
[5] `http://www.research.att.com/projects/tts`
[6] `http://www.asp.ogi.edu`
[7] `http://cnmat.CNMAT.Berkeley.EDU`
[8] `http://www.ircam.fr/anasyn`

[10] M. Wright, A. Chaudhary, A. Freed, S. Khoury, and D. Wessel, "Audio Applications of the Sound Description Interchange Format Standard," in *AES 107th convention*, 1999, www[7].

[11] D. Thom, H. Purnhagen, S. Pfeiffer, and the MPEG Audio Subgroup, "MPEG Audio FAQ," Dec. 1999, International Organisation for Standardisation, `http://www.tnt.uni-hannover.de/project/mpeg/audio/faq`.

[12] G. Peeters, S. McAdams, and P. Herrera, "Instrument Description in the Context of MPEG-7," in *Proc. ICMC*, Berlin, Aug. 2000, www[8].

[13] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth and Brooks, Monterey, CA, 1984.

[14] A. W. Black and P. Taylor, "Automatically clustering similar units for unit selection in speech synthesis," in *Proc. Eurospeech*, Rhodes, Greece, Sept. 1997, www[4].

[15] A. E. Cronk and M. W. Macon, "Optimized Stopping Criteria for Tree-Based Unit Selection in Concatenative Synthesis," in *Proc. ICSLP*, Nov. 1998, vol. 5, www[3].

[16] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, ACM Press, Addison–Wesley, New York, 1998.

[17] S. Dubnov and X. Rodet, "Statistical Modeling of Sound Aperiodicities," in *Proc. ICMC*, Tessaloniki, Greece, Sept. 1997, www[8].

[18] S. Dubnov, N. Tishby, and D. Cohen, "Hearing Beyond the Spectrum," *Journal of New Music Research*, vol. 24, no. 4, 1995.

[19] R. B. Dannenberg and I. Derenyi, "Combining Instrument and Performance Models for High-Quality Music Synthesis," *Journal of New Music Research*, vol. 27, no. 3, Sept. 1998.

[20] "Music Description Language (SMDL)," July 1995, International Organization for Standardization (ISO), `http://www.oasis-open.org/cover/smdlover.html`.

[21] C. Böhm, D. MacLellan, and C. Hall, "MuTaTeD'II: A System for Music Information Retrieval of Encoded Music," in *Proc. ICMC*, Berlin, Aug. 2000.

[22] A. W. Black and N. Campbell, "Optimising selection of units from speech databases for concatenative synthesis," in *Proc. Eurospeech*, Madrid, Spain, Sept. 1995, vol. 1, www[4].

[23] Y. Stylianou, "Removing Phase Mismatches in Concatenative Speech Synthesis," in *The 3rd ESCA/COCOSDA Workshop on Speech Synthesis*, Jenolan Caves, Australia, Nov. 1998, www[5].

[24] G. Peeters and X. Rodet, "SINOLA: A New Analysis/Synthesis Method using Spectrum Peak Shape Distortion, Phase and Reassigned Spectrum," in *Proc. ICMC*, Beijing, Oct. 1999, www[8].

[25] Y. Stylianou, "Concatenative Speech Synthesis using a Harmonic plus Noise Model," in *The 3rd ESCA/COCOSDA Workshop on Speech Synthesis*, Jenolan Caves, Australia, Nov. 1998, www[5].