MOSIEVIUS: FEATURE DRIVEN INTERACTIVE AUDIO MOSAICING

Ari Lazier, Perry Cook†

Department of Computer Science †(and Music)
Princeton University
{alazier,prc}@cs.princeton.edu

ABSTRACT

The process of creating an audio mosaic consists of the concatenation of segments of sound. Segments are chosen to correspond best with a description of a target sound specified by the desired features of the final mosaic. Current audio mosaicing techniques take advantage of the description of future target units in order to make more intelligent decisions when choosing individual segments. In this paper, we investigate ways to expand mosaicing techniques in order to use the mosaicing process as an interactive means of musical expression in real time.

In our system, the user can interactively choose the specification of the target as well as the source signals from which the mosaic is composed. These means of control are incorporated into *MoSievius*, a framework intended for the rapid implementation of different interactive mosaicing techniques. Its integral means of control, the *Sound Sieve*, provides real-time control over the source selection process when creating an audio mosaic. We discuss a number of new real-time effects that can be achieved through use of the *Sound Sieve*.

1. INTRODUCTION

Nearly all contemporary music is now produced on a computer. Much of the time, musicians record their work and then digitally edit their recordings. Recently, more and more music is created solely on the computer through the manipulation of pre-recorded sounds. In these cases, artists painstakingly choose, modify, and combine segments of sound to produce some desired effect. While this is an effective means of composition in the studio, these techniques do not function well in an improvisatory setting. They do not allow musicians to immediately respond to musical stimuli in an expressive manner. Some performers control and mix samples in real time, but are generally restricted to the standard manipulation of pitch and time, and to the application of basic time-based effects such as delay, reverb, and filtering. Recent research in Audio Information Retrieval (AIR) provides the necessary means for musicians to achieve a higher level of control over the manipulation of recorded sound. But at this time few systems have employed these techniques in the music making process.

It is currently possible to both classify sound and to retrieve audio by content with the use of a variety of features as discriminators. AIR techniques can be used to extract global information from recordings or to compare smaller segments of sound on the local level[1]. The use of these techniques to retrieve and concatenate recorded sound can be considered a form of audio mosaicing. The ultimate goal of audio mosaicing is to produce a culmulative perceptual effect desired by the artist from the combination of an ordered set of chosen segments . Thus far, this technique has generally been used to create high quality syntheses of symbolic scores or re-syntheses of existing recordings. The use of these techniques to sequence recorded sound in the absence of a score and in other interactive settings has not been explored.

This paper discusses the prospect of using audio mosaicing as a real-time means of musical expression. This can be achieved by allowing the user to interactively direct and control the mosaicing process rather than attempting to instantiate a fully specified score or other representation from a fixed set of sounds. We approach this problem by comparing the means of control of existing mosaicing techniques with the possible means of control over the mosaicing process in general. After describing how interactivity can be established by controlling different aspects of the mosaicing process, we present *MoSievius*, a framework designed to provide these means of control. *MoSievius* allows the interactive sequencing of sound based on a number of different criteria including features extracted from both recorded and live sound as well as control information provided by the user. Its overriding philosophy is to generalize the fundamental operations performed when creating mosaics in order to facilitate the implementation of both new interactive and existing offline mosaicing schemes. *MoSievius* achieves much of this control through the *Sound Sieve*, a source selection technique that can be used to apply a number of new real-time effects to both recorded and live sound.

2. AUDIO MOSAICING

Current audio mosaicing systems are derived from concatenative speech synthesis. Rather than concatenating phones or diphones as is the case with speech synthesis, the notes or sub-notes that together best correspond to some specification are combined.

2.1. Offline Target-Based Mosaicing

Target-based mosaicing is the most commonly used audio mosaicing technique. It is a descendant of image mosaicing techniques where the perceptual effect of one image, the target, is replicated with a combination of small pieces of other images. Target based mosaicing of audio signals functions similarly: a target score, represented either symbolically or as the features extracted from an existing sound, is matched with perceptually similar segments taken from some pre-chosen set of sounds.

Schwarz[2] developed a general concatenative synthesis system that worked by combining notes taken from segmented recordings. The system focuses on creating high quality syntheses of classical instruments either with reference to a MIDI score or as a re-synthesis of an existing piece of music. The matching procedure is performed with a cost function that measures both the

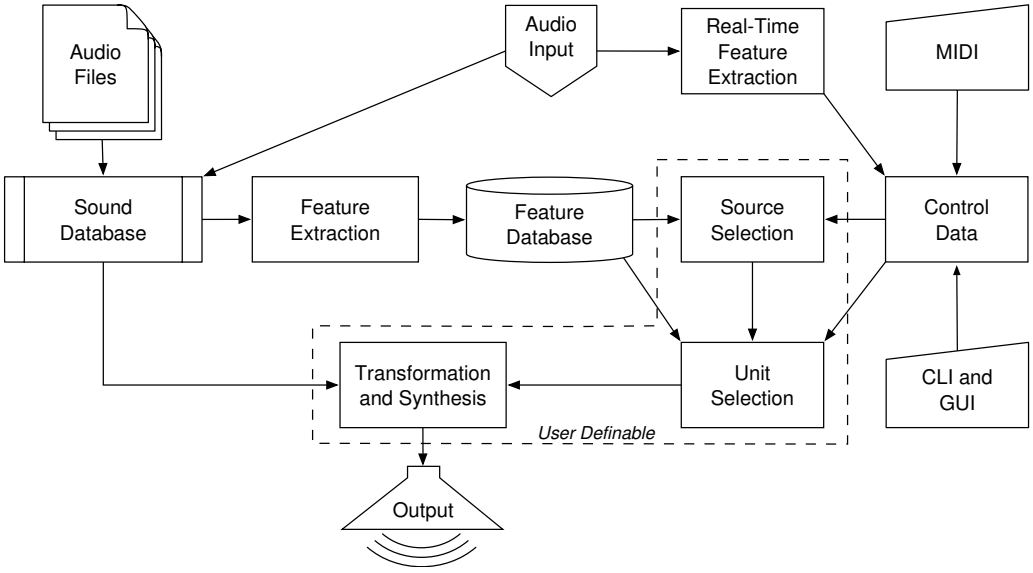

Figure 1: Mosievius

perceptual similarity between units taken from a database and target units as well as the discontinuity that would occur by choosing a particular unit. Another system for mosaicing authored by Zils and Pachet[3] uses a system of constraints to match segments of audio to a target recording. Their system used high-level features in order to direct the creation of mosaics.

2.2. Generalized Mosaicing

The creation of audio mosaics can be broken up into three components: source selection and segmentation, unit selection, and transformation/concatenation. Each component provides its own level of control over the mosaicing process. The relationship between these components can be seen in *Figure 1*.

2.2.1. Source Selection and Segmentation

The first step toward creating an audio mosaic is the selection of the initial sounds from which the mosaic is composed. For instance, if a mosaic is created solely from recordings of a trumpet, the final mosaic will sound like a trumpet, at least on the local level. Once chosen, sources are decomposed into fundamental units. Segmentation has classically been performed either on the note or sub-note (attack/sustain/release) levels. Segmentation determines the granularity and local characteristics of the resulting mosaic. Longer segments allow more characteristics of the source signals to be heard while the use of smaller segments provides more fine grained control during unit selection.

2.2.2. Unit Selection

Unit selection is the process of choosing segments to use at each point in time of the final mosaic. Units can be chosen based on any arbitrary criteria. As is the case with target based mosaicing, techniques such at retrieval by content can be used in order to find a segment of sound that perceptually matches the target unit's specification. Segments of sound can be compared to the specified features with a number of different techniques. As is the case with speech synthesis, unit selection can take into account information about previous or future segments in order to improve spectral and phase continuity at the point of concatenation [2].

2.2.3. Transformation and Concatenation

The final step consists of combining the chosen units to obtain a continuous signal. Segments can be combined with simple techniques such as overlap-add or with more complicated procedures such as pitch-synchronous overlap-add (PSOLA), that help to preserve continuity across concatenation points. Transformations such as time and pitch shifting can be performed on chosen segments before concatenation as needed.

2.3. Limitations of Offline Target Based Mosaicing

Offline target-based mosaicing schemes require fixed target and source specifications and therefore allow little interactive control over the mosaicing process. The unit selection processes of the systems described in *Section 2.1* can not be performed in real-time because they use knowledge about the target in its entirety in order to synthesize smooth transitions and to apply global constraints to the resulting mosaic. These techniques can not be generalized to real-time mosaicing where the content of target units may not be initially specified. Additionally, the systems cited above are limited by of their use of a classical representation of sound; they represent the target as individual notes or sub-notes in the form of a score or segmented recording. Therefore, the mosaics produced by these systems are restricted to the initial segmentation scheme used to represent the target.

3. INTERACTIVE MOSAICING

Any mosaicing scheme that provides a user with real-time control over the content of the mosaic can be considered an interactive mosaicing technique. Control over content is achieved during the

source and unit selection processes. The source selection process determines the content from which segments are chosen while the unit selection process determines which segments are to be used at each particular point in time. The concatenation phase involves modifying the chosen content in order to obtain a better result.

As long as the user has real-time control over either source or unit selection, the technique can be considered a form of interactive mosaicing. Even if a target has been fully specified, the ability to interactively change the sources from which segments are chosen in real time provides the user with interactive control. Similiarly, if source selection is fixed then the user can attain interactivity through real-time specification of the target segments.

3.1. Real-Time Target Specification

Interactivity can be established in target-based schemes by permitting the user to dictate parts of the target as needed rather than requiring the initial specification of the target in its entirety. Target units can be specified in real-time as the desired features of each segment. As is the case in offline mosaicing, these features can then be used in retrieval by content to choose a segment that exhibits the specified qualities. There are multiple ways to specify the desired features of a segment. One method is to extract features from an input audio signal. A more direct method is to allow the user to dictate the features of each target segment. This can be done either by manually specifying feature values or by specifying the target symbolically with MIDI or some other notation. If a symbolic specification is used, a mapping between the symbolic description and its equivalent in terms of the features used must be found. For example, vibrato can be mapped to changes in pitch and energy over time. To add vibrato to a segment, the changes in pitch and energy over time that correspond to the desired amount of vibrato can be incorporated into the segment's feature represention.

One use of real-time target specification is the replacement of a sound captured in real-time will similar sounds from a different source. In this case, target segments are specified by the features extracted from the live source while the segment search space consists of previously recorded sounds. For example, features can be extracted from a trumpet in real time and used to find the closest corresponding sounds in a recording of a trombone. The trajectory of different segments' MFCCs can be used to calculate the perceptual similarity between the target segment and trombone segments [4]. Techniques such as Radial Basis Transformation can be used to warp the extracted trumpet features in order to obtain a better match against the trombone [5]. Because the trumpet player is specifying the features of each target in real-time, he possesses interactive control over the system.

This technique has a best-case latency equal to the size of the target segment used in unit selection. This is the case because the features of an entire unit need to be known before a source segment can be chosen. There are two ways to reduce this latency. The simpler solution is to choose the segment that matches best at any particular point in time. Once the incoming signal differs a sufficient amount from the chosen source segment, a new source segment can be chosen. This is one instance where interactive control over segmentation proves to be quite useful. In this case segment boundaries are only defined once the entire segment has been played.

The second solution involves using small enough segments that the latency become perceptually insignificant. This solution requires the ability to synthesize high quality transitions between incongruent source segments. Such problems are dealt with in concatenative speech synthesis by accounting for spectral and phase discontinuity across segment boundaries during both unit selection and concatenation [6, 7]. Many of these techniques are designed to work across phone or diphone boundaries and must be generalized to work across segment boundaries for other types of signals.

3.2. Interactive Source Selection and Segmentation

Another means of control over interactive mosaicing is through the manipulation of the source selection process. This can be done while using a fixed target or when specifying a target in real-time. When using a fully specified target, the target can either be used directly or manipulated over time for added control. Interactive source selection can be added to the example of re-synthesizing one instrument with sounds of another by removing the restriction of the source sounds to those of a single instrument. If sources from multiple instruments are used, the user can specify the sound set to use at each point in time. This allows the choice of instrument for the source segments to change during the creation of the mosaic. The use of sub-note segmentation allows midnote transitions between different sound sets.

In such situations, warping the features of the target enables smooth transitions between multiple sound sets and the creation of "hybrid" instrument sounds. For example, one can create a mosaic that sounds like a cross between a saxophone and a flute. Even though the sources used come from either a saxophone or a flute, warping the features correctly will ensure that the chosen saxophone sounds have "flutish" qualities while the flute sounds chosen have some qualities of a saxophone. Consider the case where a saxophone solo is used as a fixed target while the source segments consist of recordings of a flute. If the features from the target are not warped to reflect the qualities of a flute, then the resulting mosaic will consist of flute sounds that sound like a saxophone.

With this type of mosaicing the problems encountered in real-time target specification will still occur, but to a lesser extent. If the entire target is known, then the techniques used to minimize discontinuities at the concatenation points can be used. But these techniques do not yet synthesize flawless transitions between segments even when the all segments are taken from a single source. These techniques need to be further refined to reduce the number of artifacts and discontinuities to a level acceptable for real-time applications.

3.3. The *Sound Sieve*

In the example given above, the source instrument, an attribute that was hand labeled, was used to interactively control the source selection process. It is more interesting to be able to do this for any arbitrary signal where, rather than using sources that are known to come from a flute, other sounds which are perceptually similar to a flute are used. Using techniques such as instrument identification can be used as a replacement for hand labeling. The possibility that a sound came from a particular instrument can be found by comparing a segment's cepstral coefficients to clustered sets of cepstral coefficients extracted from recordings of different instruments [8]. When using these probabilities as features, a final decision on the identity of the instrument for each segment does not need be made. Instead the probabilities of a sound coming from different sources can be used directly as a means of defining a sub-space of source

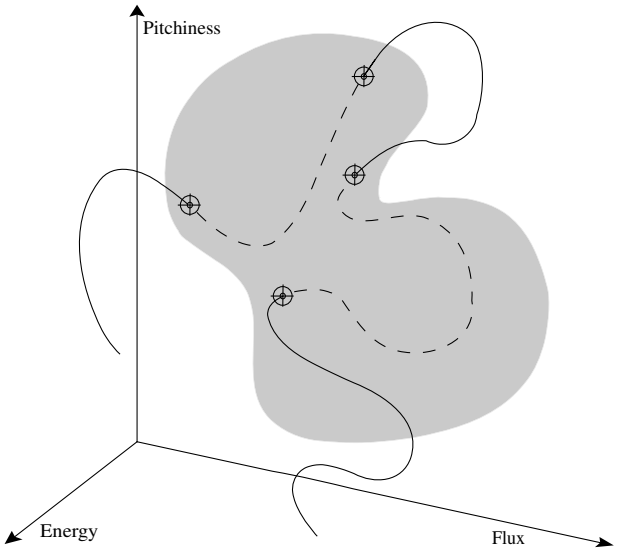

Figure 2: Three Feature Sieve

segments. Instead of warping the target to exhibit specific qualities, here the search space is limited to segments which have those qualities. The same effect can be obtained without warping the features of the target segment through greater control over the source selection process.

The process of isolating a sub-space of segments can be generalized to any set of features. We have named this means of source selection the *Sound Sieve*. The *Sound Sieve* uses features extracted from audio signals in order to limit the unit selection search space from a set of segments to a subset of those segments described by their location in feature space. By finding mappings between features and their perceptual equivalents, it is possible to use the *Sound Sieve* to isolate segments of a recording that exhibit certain characteristics.

A simple means of isolating a sub-space is by setting thresholds or ranges for each feature. The segments within the specified ranges will pass through the sieve. This concept is illustrated in *Figure 2* where three features, pitchiness, RMS energy, and spectral flux are used to segment a single source. The shaded area is the subspace defined by the feature ranges or thresholds, and the line is the time-varying trajectory of the features over the duration of the recording. The dotted lines are segments that are within the chosen sub-space and pass through the sieve while the solid lines represent segments that lie outside of the chosen space.

The process of *Sieving* is analogous to the "blind" segmentation performed by Tzanetakis [9]. In addition to finding segment boundaries, the *Sieve* puts each segment into one of two categories, those that pass through the *Sieve* and those that do not. If the right features and ranges are chosen, one can use the *Sieve* to separate notes from silence, vowels from consonants (see *Figure 3*), to find the segment boundaries between steady and transient segments of a single note, or to isolate any class of segments that can be described by some relation within the feature space.

3.3.1. Source Selection with the Sieve

One use of the *Sound Sieve* is as a means of restricting the search space from which segments are chosen when creating a mosaic from a fixed target. This involves replacing segments taken from an existing recording or a score with their closest counterparts in a set of sieved segments. This technique functions as a more elegant solution to the problem of "bad segments" encountered by Schwarz where segments with undesirable qualities were chosen during unit selection [2]. Instead of marking individual segments for removal from the search space, it is possible with the sieve to weed out entire classes of segments during run-time that can be described by some relationship between the bad segments' features.

This also provides a new level of control over the qualities of the resulting mosaic; it restricts the choice of segments to those with the desired qualities, either ruling out undesirable segments or emphasizing specific qualities in the final mosaic by limiting the search space to specific segments. The sieve allows the user to interactively modify the creation of a target-based mosaic by changing the set of segments from which the unit selection algorithm chooses at any point in the mosaicing process. This can be used to change the textural or timbral qualities of the segments during the construction of the mosaic.

3.3.2. Target Manipulation with the Sieve

Although the *Sieve* can be used as a means of obtaining a set of segments to be used in conjunction with target based mosaicing, it can also be used as a unit selection process in and of itself. One way to do this is to sieve an existing target and to concatenate the segments that pass through in the same order they are taken from the original target. This procedure combines the tasks of source and unit selection into one. In essence, the choice of a unit at each particular point in time is determined by the sources that pass through the sieve. Because every segment that passes through the sieve is heard in the final mosaic, this technique allows one to "hear" the perceptual qualities of different parts of the feature space.

The use of time shifting can be used to preserve the overall temporal structure while changing the content of a recording. As an example, when this technique is applied to speech signals, changing the thresholds to allows only non-pitchy (i.e. noisy) segments to pass through extends the length of consonant sounds while shortening and eventually eliminating the more pitchy vowel segments. This is illustrated in *Figure 3*. The top spectrogram was generated from a male speaker reciting the phrase, "This is an example of male speech." Ranges for spectral flux and pitchiness were used to isolate vowels and consonants. Segments which passed through the sieve were time shifted in order to maintain their original location in time.

The chosen sub-space can also be altered over time to achieve a number of other effects. It is possible to alter manually the range for one feature while automatically expanding and contracting the ranges of other features so that a constant number of segments always pass through the sieve. As the chosen subspace moves, segments that remain *chosen* change in temporal location. Playing a file in a loop while systematically changing the feature ranges can be used to create a number of interesting rhythmic effects. For instance, with a quantitative measure of "beatiness", it is possible to automatically change feature ranges in order to produce beats at specific points in time.

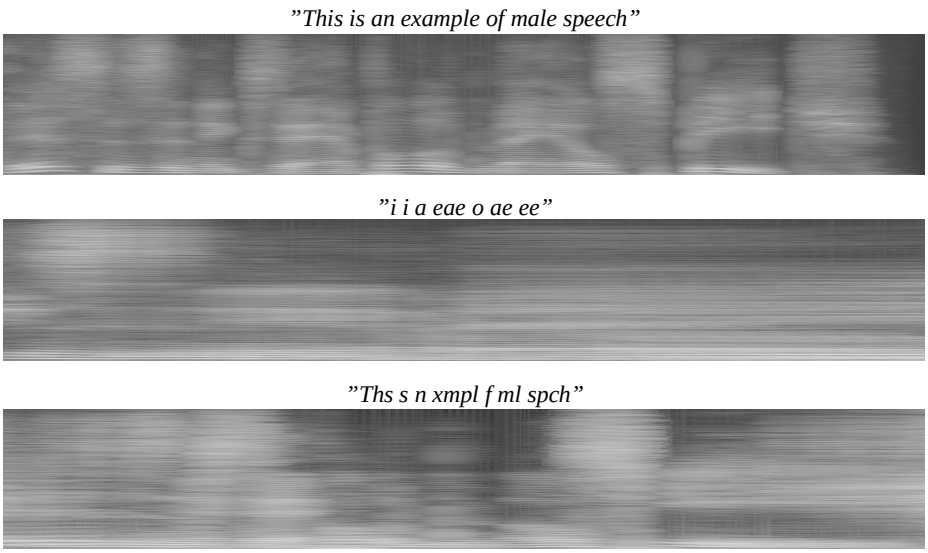

Figure 3: Vowel/Consonant Isolation with the *Sound Sieve*

4. MOSIEVIUS

The purpose of *MoSievius* is to faciliate the implementation and experimentation with interactive mosaicing techniques. The main objective of the system is to present a unified interface that allows the user to interactively control source selection, segmentation, unit selection, and synthesis. The system does not implement one particular mosaicing technique, but provides an interface to many of the common operations performed when sequencing sound. In short, it is a centralized framework with the functionality to build real-time mosaicing applications such as those described in *Section 3*. *MoSievius* can be used as the following:

- A c++ library for building real-time mosaicing applications.
- An "out of the box", standalone interactive audio mosaicing engine. It includes all of the modules in *Figure 1*, MIDI and other io support, a custom built interpretive language (*MohAwk*) for writing interactive mosaicing programs, and various GUIs for controlling *MoSievius*.
- A backend to extend existing audio programming environments (via plugins).

The *MoSievius* engine transparently handles feature extraction, segment manipulation and storage, retrieval by content, and transformation and synthesis in response to high-level requests made by the user.

4.1. Overview

An overview of the system can be seen in *Figure 1*. Features are automatically extracted from sound files once they are added to the sound database by the user. Once a sound is added, it can be used as a target for unit selection or as a source to be used in the mosaic. Real-time audio can also be added to the database and used for these purposes. The three controllable aspects of the mosaicing process, source selection, unit selection, and transformation and synthesis are implemented by the user. Because the user is given complete control over these processes, the user can direct the mosaicing process in reference to features of any sound in the database and features extracted from input audio, as well as control data such as MIDI messages and messages from the command line (CLI) or GUI. Once the user chooses a segment to be played, he instructs the *MoSievius* engine to play that segment along with the effects and transformations which should be applied as well as the technique which should be used for concatenation.

4.2. Representation

All segments of sound in *MoSievius* are represented identically as an ordered set of sub-segments. The smallest level segments are individual windows that are taken from the analysis phase. This representation was chosen in order to allow multiple levels of segmentation. Each level can be labeled with a user definable tag based on the type of segmentation performed. As an example, a signal can first be segmented by note, and then by sub-note to obtain transients and steady parts of the note. A user can then retrieve an attack with particular qualities or a note that contains an attack with those qualities from the database. Access is provided to sub-segment as well as global segment features.

This representation also allows every sound in the system, including sound which is captured in real-time, to be represented identically. All sounds or segments in the database can be used interchangeably during source selection, unit selection, or synthesis. For example, this representation allows a user to play back modified versions of sound that other musicians have produced. It also allows one to use features extracted from live audio as a target specification or as a means of controlling source and unit selection. This allows a *MoSievius* user to sieve live sound or to perform context-aware mosaicing where the system is aware of the sounds being produced by other musicians.

4.3. Implementation

MoSievius currently runs on Mac OS X, Linux, and Windows. There are two main components to our implementation. The audio

information retrieval engine and the effects and concatenation engine. All feature extraction is performed with the *Marsyas* framework [9]. An implementation for retrieval by content using a fast k-nearest neighbor search with the use of arbitrary sets of features is provided. A number of effects such as time and pitch shifting are implemented and optimized to minimize latency. When playing segments, a user can specify whether the engine uses overlap/add or PSOLA.

The current implementation can be used in a number of different ways. The most straightforward is to write individual mosaicing programs. This allows the use of custom GUIs or other forms of communication as a front end to the system. *MoSievius* can also be used in combination with interpretive audio programming languages for added control. We have written *MohAwk*, an audio enabled version of the Awk programming language which allows the rapid prototyping of different mosaicing algorithms. *MohAwk* leverages the callback structure of Awk to provide an elegant (although bizarre) and fast means of implementing MIDI controllable mosaicing algorithms.

5. CONCLUSION AND FUTURE WORK

We have discussed the process of interactive mosaicing and have explored a number of new applications and effects which can be achieved through these new techniques. MoSievius is still being refined, and we look forward to discovering and experimenting with even more new mosaicing techniques. New features are currently under development, as well as new effects and concatenation techniques. A number of new means of controlling MoSievius are also being developed. A MoSievius patch for Max/MSP is under construction. Also, the inclusion of MoSievius into other real-time languages such as *Chuck* [10] is planned. We are considering the creation of MoSQL, as a simple means for feature-based sample storage and retrieval. *MoSievius* is freely available at:

`http://soundlab.cs.princeton.edu/research/mosievius/`

6. REFERENCES

[1] Jonathan Foote, "An overview of audio information retrieval," *ACM Multimedia Systems*, vol. 7, no. 1, pp. 2–11, January 1999.

[2] Diemo Schwarz, "A system for data-driven concatenative sound synthesis," in *Proc. of the COST G-6 Conf. on Digital Audio Effects (DAFX-00)*, Verona, Italy, 2000.

[3] Aymeric Zils and Francois Pachet, "Musical mosaicing," in *Proc. of the COST G-6 Conf. on Digital Audio Effects (DAFX-01)*, Limerick, Ireland, 2001.

[4] Christian Spevak and Emmanuel Favreau, "Soundspotter - a prototype system for content-based audio retrieval," in *Proc. of the COST G-5 Conf. on Digital Audio Effects (DAFX-02)*, Hamburg, Germany, September 2002.

[5] Carlo Drioli, "Radial basis function networks for conversion of sound spectra," in *Proc. Cost-G6 Conf. on Digital Audio Effects (DAFX-99)*, Trondheim, Norway, 1999.

[6] Esther Klabbers and Raymond Veldhuis, "Reducing audible spectral discontinuities," *IEEE Transaction on Speech and Audio Processing*, vol. 9, no. 1, January 2001.

[7] Yannis Stylianou, "Removing phase mismatches in concatenative speech synthesis," in *The 3rd ESCA/COCOSDA Workshop on Speech Synthesis*, Jenolan Caves, November 1998.

[8] Judith Brown, "Computer identification of musical instruments using pattern recognition with cepstral coefficients as features," *J. Acoust. Soc. Am.*, vol. 105, pp. 1933–1941, 1999.

[9] George Tzanetakis, *Manipulation, Analysis, and Retrieval Systems for Audio Signals*, Ph.D. thesis, Princeton University, June 2002.

[10] Ge Wang and Perry Cook, "Chuck: A concurrent, on-the-fly audio programming language," in *Proceedings of the International Computer Music Conference (ICMC)*, Singapore, September 2003.