CONCATENATIVE SYNTHESIS
MIR Final Project

Fall 2012

Amar Lal and Kevin Murphy

1. INTRODUCTION

Increasingly large libraries of music exist on our hard drives, as both older music is digitized and newer music is made. As composers in the 21st century begin exploring new sounds in the same "opening up of music to all sounds" (coined by Chadabe) fashion as composers in the 20th century, many remix and mashup artists have already begun to take advantage of the wealth of sounds available within these ever-growing databases. However, as addressed by Zils and Pachet [9], the manual search and retrieval of appropriate samples becomes exponentially more difficult as the size of a database increases. Samples are often not classified, especially on a deeper musical level, although databases such as the Million Song Database attempt to begin solving this massive problem.

In addition, generating meaningful sequences from these large libraries can also be very difficult; to arrange material that contains such a dense amount of information in such a way that the sequence is cohesive on at least a few levels of musical detail requires an attention to detail and capacity for problem solving that is often above the capabilities of most humans.

For this reason, systems of "musical mosaicing" have been proposed and implemented, wherein sounds are chosen and extracted from a database automatically by a program, based on target characteristics extracted or specified from a target score. This score can be a sequence of MIDI notes or other information, or audio from which to extract certain characteristics.

In his work [7], Schwarz proposes a "corpus-based" system for concatenative synthesis, wherein a database (the corpus) of descriptors is built up from the analysis of a sound library and stored for later access in the unit selection process.

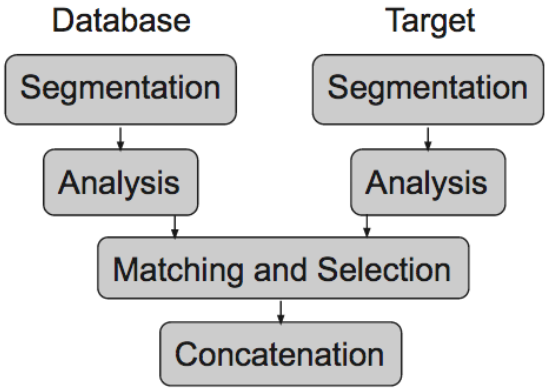
In the work of Jehan [4], music is segmented by "events", as marked by onsets. The audio examples associated with the work are among the more compelling examples of musical mosaicing that are readily available; they maintain a more musical sensibility, as music is generally considered an event-based organization of sound.

1.1 Framework

For this reason, we chose to implement an event-synchronous system, where a library of sound files is first segmented into "events" based on their onsets before each event is analyzed in a various number of ways. The start and end times of these events along with the analysis descriptors are stored in a corpus. The target audio file is segmented and analyzed in a similar way, and descriptors are stored in a separate data structure. A distance calculation between each target sample and each sample of the database is used to match and select appropriate samples from the database for concatenation. Before being concatenated, the samples are time-stretched using a phase vocoder based on Dan Ellis' implementation [10]. The selected samples are then concatenated in order to form an output file.

The block diagram for this framework can be seen in Fig 1.

Fig 1: Block diagram framework for the proposed system



1.2 Goals

It is our hope that we can begin to explore the compositional possibilities available with this system. As discussed by Schwarz in [7], the broader hope is to "give rise to new sonorities and new methods to organize and access them, and thus expand the limits of sound art". Zils and Pachet also point out the wonderful new ability to listen at "two levels" in [9]; one level where the listener examines the output which sounds similar to a target, and one level which reveals the different samples that were used to create the final output. Hopefully this system will be robust enough to begin working with small libraries and determine ways to expand and improve upon the system both based on the work of others and through findings from trials.

2. SEGMENTATION

It is our hope that the practice of segmenting both the database and target files into event-synchronous samples will yield more musical results. Jehan discusses this in his work; "from an auditory scene analysis point of view, by which humans build mental descriptions of complex auditory environment, an abrupt event is an important sound source separation cue" [4]. By this rule, one would infer that by sequencing a series of abrupt events together (which may contain smaller sequences or phrases themselves), it would be possible to create a sequence of sound events that could be deemed "musical".

Additionally, by concatenating segments at the strategic location of an onset, we avoid the need for phase shifting or crossfading samples to create artifact-free transitions. Although we do not go to the level of the work of Jehan, wherein sample start times are moved to a zero crossing that precedes the onset, we still have found the output of our implementation to be relatively artifact-free.

2.1 Complex Domain Onset Detection

We chose to use a complex domain onset detection function for our work. Although this method is slightly more computationally expensive than the simple spectral magnitude prediction used in the work of Jehan, we have found it to be a robust system, as it has been found to be more

sensitive to both tonal and percussive sounds, as discussed by Davies in [11]. This function takes into account both a spectral magnitude and phase prediction based on 3 frames of the signal's FFT. The resulting function is post-processed, first by half-wave rectification, by normalization, by low-pass filtering with a butterworth zero-phase reverse IIR filter (filtfilt) of order 10 and normalized cutoff frequency 0.09 (around 3.6kHz, to allow sharp transients). The result is again half-wave rectified, the local energy is taken by squaring the signal, and the result is normalized. The detection function is peak-picked with a window of ~2ms to eliminate any double-picking of noisy transients. A user-specified fixed threshold is available, but was not used in our implementation to allow for signals of many different levels.

Fig 1.1 Shows the complex domain function's performance for a 12-second rhythmic sample.

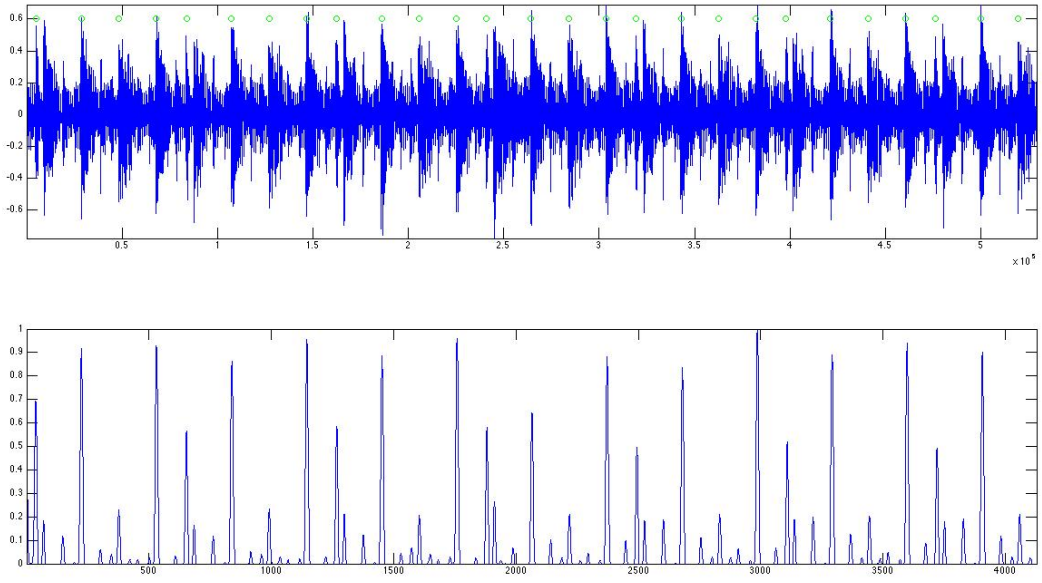


Fig 1.1 Complex Domain detection function (below) and detected peaks (green circles above) for rhythmic signal.

2.2 Start and End Times

These extracted onset times were used as sample indices for the start and end times of each segment. Pairs with an overlap of one value were created using the *buffer* function, to create a 2XN matrix of start and end times (where N is the number of segments). These start and end times are stored in the corpus for use in selecting blocks of audio both for analysis and for concatenation.

We had initially considered using a beat-tracking function to align the onsets more accurately by tempo, but, as is possible to see in Fig 2, the detection function was able to extract a steadily linear sequence of start and end times from a rhythmic signal, so this extra process was deemed temporarily unnecessary.

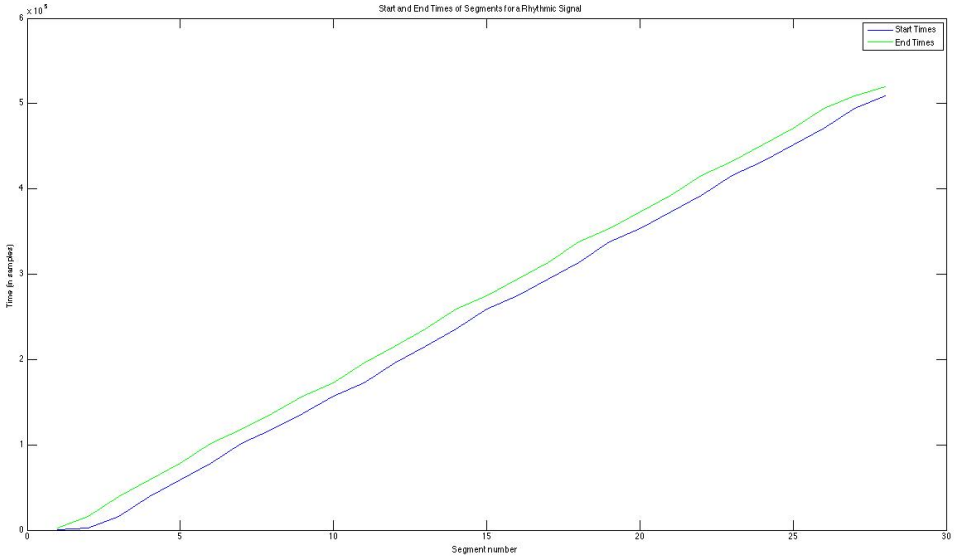


Fig 2: Start and end Times for a rhythmic signal. Start times in blue, end times in green. It is possible to see the linear relationship both between the two and between each line and time, denoting a relatively steady or rhythmic sequence of samples in time.

3. ANALYSIS

We chose several parameters for analysis of audio signals based on some basic concepts we deemed most important in music. When writing music, composers usually pay attention to pitch, dynamics, and timbre (or instrument) as basic constraints. To address pitch, we chose chroma and maximum chroma value, which would address the "root note" of the sample as well as possible chords or polyphony. To address dynamics, we chose the rms amplitude of the signal as a measure of overall loudness, and the maximum amplitude as a safety for sharply transient or amplitude-varying signals. To address timbre, we chose the basic measure of spectral centroid, as it may be difficult to characterize timbre in a more detailed fashion for complex signals.

3.1 Chroma

Our Chroma function uses a constant-Q comb filter approach, and a base pitch of 55 Hz. 4 octaves above this base pitch are analyzed at a resolution of 12 bins per octave. For the purpose of storing smaller amounts of information, we took the mean of the chroma over time for each sample, and used this as the chroma vector for the sample. We also extracted the maximum chroma value as a measure of root or base note of the sample. Fig 3 shows the chroma function analyzing a C major scale as played by a piano. Fig 4 shows chroma values for a series of chromatic sine tones.

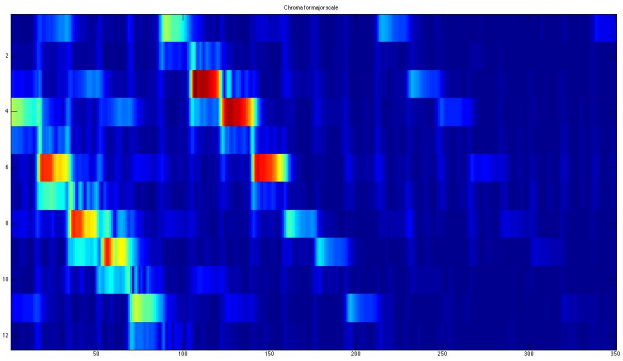


Fig 3: Chroma values for a major scale played by a piano. Note the order of whole and half steps is preserved.

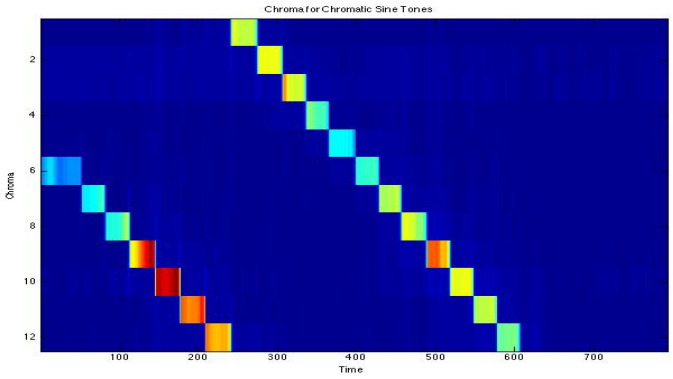


Fig 4: Chroma values for a series of chromatic sine tones

3.2 Spectral Centroid

Spectral centroid is used to characterize the center of mass of the spectral distribution. This is often used to characterize the timbre of brightness of the signal. It is calculated as a frequency-weighted mean of the power spectrum of the signal. Fig 4 shows the spectral centroid of chromatic sine tones, with green lines representing actual chromatic pitches.

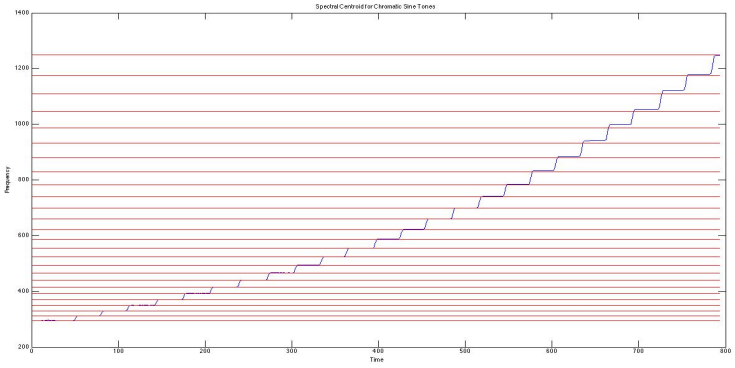


Fig 4: Spectral Centroid for Chromatic Sine Tones. Horizontal Lines represent chromatic pitch values.

3.3 Amplitude

We chose RMS amplitude as a fair single measure of the amplitude of the sample. In order to account for any possible perceived loudness from sharp transients or bursts of energy, we also chose to include the maximum amplitude of the sample in our corpus. In future implementations, it might also be interesting to include an amplitude envelope measure of some kind, or a standard deviation from the RMS level.

4. MATCHING AND SELECTION

Once the audio signals have been segmented, analyzed, and their descriptors have been stored in a database, the actual sequence of events is generated. This is done by matching each of the target samples to a most similar sample from the database.

4.1 Distances

First, a basic distance between the target feature and each feature in the database is calculated. This is done on a signal-by-signal basis, so that the distances can be calculated more efficiently as vector operations on the corpus vectors for each signal (i.e. distance between target RMS amplitude and the vector of RMS amplitudes for each sample in a given file). For RMS and maximum amplitude, simple magnitude differences are taken. For maximum chroma and spectral centroid distances, a normalized magnitude is taken. For chroma vectors, a Euclidean distance is taken (as implemented in *AL_EDist.m*).

4.2 Segmentation Cost

Once all of these distances has been calculated, the challenge becomes how to combine them into some sort of meaningful single value that represents each. In the literature, this value is referred to as a "segmentation" or "target" cost, or the cost of picking a specific segment based on given constraints. Segment costs are generally calculated as a weighted sum of the feature distances, allowing the "composer" to give priority to certain features when calculating distances. These features and their weighted combinations act as constraints on the composition, and, as Schwarz explores in [8], these constraints can often be useful when composing. To weight the features, we have supplied an optional distance vector parameter, which allows the composer to weight any of the feature as they see fit. If this parameter is unspecified, the weightings are all equal (set to 1). Effects of different weightings will be explored in section 6. The order of the weighting vector is RMS amplitude, Max Chroma, Max amplitude, Chroma, Spectral Centroid.

4.3 Concatenation Cost

The literature also often refers to "concatenation cost", which is the "discontinuity introduced by concatenation of the unit from the database with the preceding unit" [8]. In other words, it's a measure of the similarity or difference between the current chosen sample and the previous chosen sample. We have not implemented this concatenation cost in our work, choosing

instead to begin our exploration of concatenative synthesis by focussing on the effects of segmentation costs.

5. CONCATENATION

Many systems, such as Schwarz' [8], apply transformations to the selected sample in order to make their features more similar to the target. This is often done because the "musaicing" is done at much smaller time levels, such as frames of audio. We have chosen not to implement any changes except time shifts to keep the selected samples within the event pattern of the target, in an effort to maintain any rhythmic or phrasal aspects of the target. To do this, we used a phase vocoder. After phase vocoding, samples are simply placed in the appropriate section of an output buffer, which is written as an output wave file.

5.1 Phase Vocoding

We did not implement a phase vocoder of our own, as this was not the focus of the work. We chose instead to use a modified version of Dan Ellis' MATLAB implementation [10]. The ratio of time shift is given by the ratio of the length of the target sample to the length of the selected sample. Because the phase vocoder does not always shift to the exactly right number of samples, we added two safeties after phase vocoding to ensure the sample is the intended length. If the phase vocoded sample is longer than the target, the extra samples are simply discarded. If the phase vocoded sample is shorter than the target, the sample is simply zero-padded to the desired length. There are likely more musical ways of fixing this time discrepancy, but, as previously mentioned, this transformation phase was not the focus of the work, so we did not spend much time experimenting with other methods.

6. RESULTS

In order to evaluate our unit selection function, we used a similar methodology to that of Zils and Pachet [9], where certain features are weighted fully high and all others are not included. This allows us to examine each feature individually. We chose to examine chroma and spectral centroid. Each of these tests were generated using the same chromatic sine wave audio signal used to generate the spectral centroid figure above (Fig 4). The library used consisted of a performance of the Beethoven piano sonata No 3 Mvmt 3, a performance of the Bach Cello suite No 1, Prelude, a solo clarinet piece, a solo flute piece, a solo marimba piece, and the Beatles' 8 Days a Week.

6.1 Chroma only

To test the chroma function, we weighted the chroma to 1 and all other distances to 0. The results are seen in Fig 5 below.

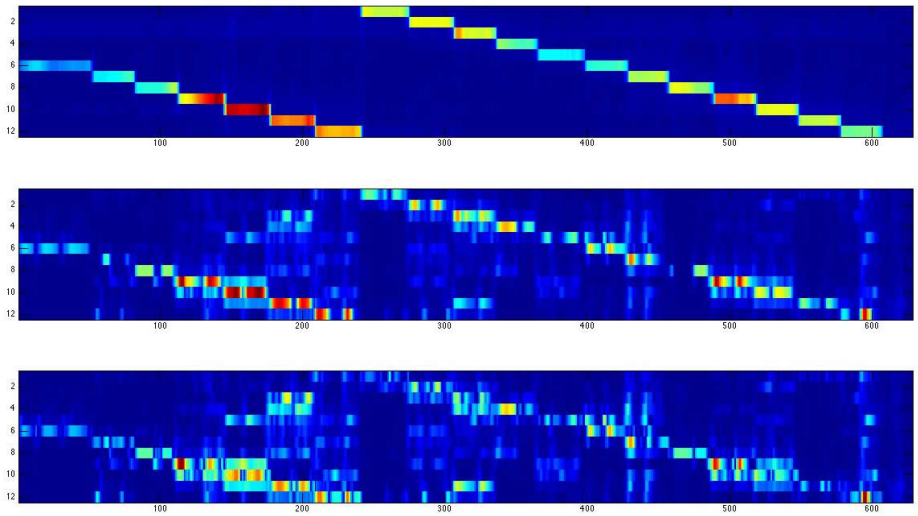


Fig 5 Chroma values for (1) Chromatically increasing sine wave (2) concatenation based on sine wave target (3) difference between chroma matrices

It's possible to see that the concatenated version follows the general path of the chroma, with some expected added noise from polyphony and other sounds. There appears to only be one gap, which may be attributed to the small size of the library. The difference between the original chroma and output chroma appears to show most of the noise in the signal, with a smaller focus on the original path, showing that there was a pretty decent correlation between the two. Audio sample is titled 'ChromaOnly.wav'

6.2 Spectral Centroid only

The test for Spectral Centroid only is below. We used the chromatically increasing sine waves for this.

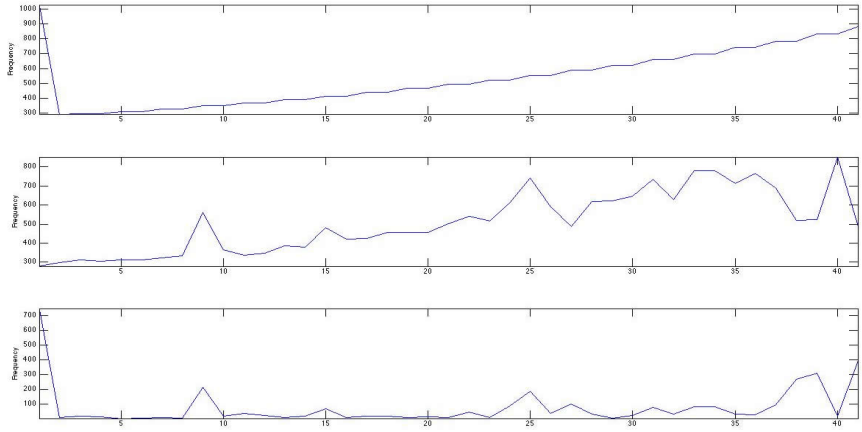


Fig 7: Spectral centroid for (1) Chromatically increasing sine wave (2) concatenation based on sine wave target (3) difference between vectors

It's possible to see that the concatenation follows the general shape of the target sequence, again with a little bit of noise likely due to the small size of the database used. Audio sample is

titled ‘SpecCent.wav’

6.3 Unweighted Sum

For this test, we used Bob Dylan’s “Blood on the Tracks” record as the library, attempting to recreate the first minute of Tangled up in Blue using all of the other tracks on the record. Audio sample is title ‘unweightedDylan.wav’.

Chroma actually works surprisingly well, and spectral centroid and max chroma seem to work well within certain reason. The output has some actual chordal similarity to the original, and seems to sound somewhat cohesive, if a little novelty.

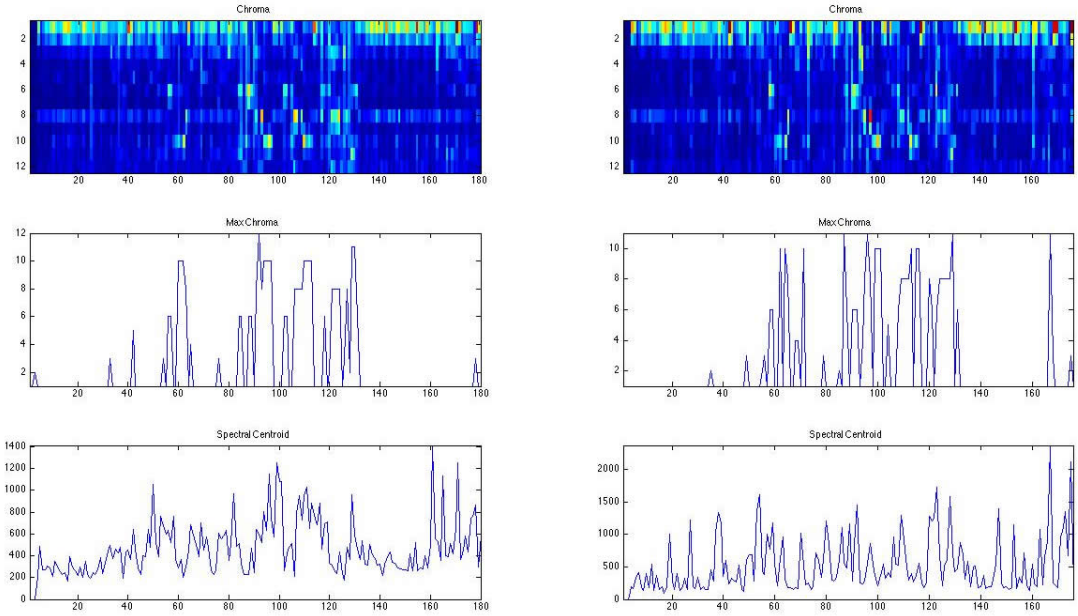


Fig 8: Rows: (1) Chroma (2) Max Chroma (3) Spectral Centroid
Columns: (1) Target (2) Output

7. FUTURE WORK

Although the output of this implementation is definitely interesting, it is still more novelty than musical, as the listener enjoys the strange sequence of samples generated. We are only accessing one of the two levels of listening discussed by Zils and Pachet, the level of the individual samples used in the sequence. The output is not yet musically meaningful enough to access the second level of listening (arguably more difficult), in which the user is able to listen to the output’s similarity to the target track - although we do not seem too far off.

In order to make this output more coherent, it would be useful to examine the implementation of concatenation costs, as discussed in [2],[9],[8]. Hunt and Black also discuss the use of the Viterbi algorithm in finding the lowest concatenation cost, which would make for an efficient implementation [2]. Along with Schwarz [8], they also discuss the idea of training the cost functions on data with linear regression both to speed up the search process and to generate more meaningful weightings for the segmentation cost sums.

It may also be useful to explore different ways of representing features for each sample, with their mean, standard deviation, min/max, slope and normalized spectrum, as discussed in [8]. This seems somewhat like overkill for many features, but it would be useful to explore the most pertinent of these to describing the "shape" of the features over time for each sample.

The current implementation of our concatenative synthesizer chooses samples that are the most similar (least euclidean distance). However, it might be musically interesting to choose samples are that are the least similar (largest euclidean distance). This implementation may give a result that sounds like an inverse concatenative synthesis of the original audio file.

All of this would add to a system that is already relatively slow, so it would be useful to implement a k-nearest neighbours approach as in [2][8], to speed up the distance calculation and search processes.

8. REFERENCES

[1] "Concatenative Synthesis Using Score-Aligned Transcription." *Concatenative Synthesis*. 16 Nov. 2012. <http://www.cs.cmu.edu/~music/concat/concat.html>.

[2] Hunt, Andrew J., and Alan W. Black. *Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database*. ATR Interpreting Telecommunications Research Labs.

[3] Janer, J., M. Haro, G. Roma, T. Fujishima, and N. Kojima. "Sound Object Classification for Symbolic Audio Mosaicing: A Proof-of-Concept." *Music Technology Group*. 23 July 2009. 16 Nov. 2012. <http://mtg.upf.edu/node/1360>.

[4] Jehan, Tristan. "Event-Synchronous Music Analysis/Synthesis." *DAFx'04*. Proc. of 7th Int. Conferenence on Digital Audio Effect, Italy, Naples. 2004.

[5] Lin, Heng-Yi, Yin-Tzu Lin, Ming-Chun Tien, and Ja-Ling Wu. "Music Paste: Concatenating Music Clips Based on Chroma and Rhythm Features." *National Taiwan University* (2009): *ISMIR*. 16 Nov. 2012. <http://http://ismir2009.ismir.net/proceedings/PS2-4.pdf>.

[6] Schwarz, Diemo. *Distance Mapping for Corpus-Based Concatenative Synthesis*. IRCAM-CNRS-UPMC.

[7] Schwarz, Diemo. *The Sound Space as Musical Instrument: Playing Corpus-Based Concatenative Synthesis*. IRCAM-CNRS-UPMC.

[8] Schwarz, Diemo. "A System for Data-Driven Concatenative Synthesis." *DAFX-00*. Proc. of COST G-6 Conference on Digital Audio Effects, Italy, Verona. 2000.

[9] Zils, Aymeric, and Francois Pachet. "Musical Mosaicing." *DAFX-01*. Proc. of COST G-6 Conference on Digital Audio Effects, Ireland, Limerick. 2001.

[10] Ellis, Dan. *A Phase Vocoder in MATLAB*. Columbia University, MATLAB. <http://www.eumus.edu.uy/eme/ensenanza/electivas/dsp/presentaciones/ejemplos_clase/phase_vocoder/A%20Phase%20Vocoder%20in%20Matlab.htm>

[11] Davies, M., and M. Plumbley. *Beat Tracking with a Two-State Model. Queen Mary, University of London*. Centre for Digital Music,