Cepstral Analysis Tools for Percussive Timbre Identification

William Brent
Department of Music and Center for Research in Computing and the Arts
University of California, San Diego
wbrent@ucsd.edu

ABSTRACT

A set of low level and cepstral feature extraction externals for pure data (Pd) are introduced and evaluated in a percussion instrument classification problem. Because this set of tools is intended for immediate real-time timbre identification using an accompanying classification external, a constraint of reporting results within 30 ms after a detected onset is applied. While Bark-frequency cepstrum is the most effective single feature, it is shown that combining several simple measures into a feature vector produces useful results as well. Temporally evolving feature data collected by taking a series of successive analysis snapshots rather than a single snapshot immediately after onset detection is also discussed.

Keywords

Cepstrum MFCC Barks Timbre

1. INTRODUCTION

The relationship between subjective timbre judgments and deterministic acoustic measurements is a research problem with a rich history. [2] [3]. The classical experiment in this area sets out to record timbre similarity judgments from a group of subjects and then feed the data to a multidimensional scaling (MDS) algorithm. Based on the results of MDS, instruments are assigned coordinates in a low dimensional space (typically 2 or 3), where geometric proximity in the space implies timbral similarity. At that point, the challenge is to find objective acoustic measures that correlate strongly with the grouping of instruments along any of the axes. Two measures that consistently reappear in this type of research are spectral brightness and log attack time.

Several other measures are currently being used as timbre features in the area of music information retrieval, including spectral brightness, spectral flatness, spectral roll-off, spectral flux, spectral centroid, zero crossing rate, and mel-frequency cepstral coefficients (MFCCs) [4]. In some applications, these features can be tracked in detail by taking successive short-time analyses over the course of an entire sound event.

A great deal of further research is needed to clarify relationships between these measures and the perception of timbre. However, basic feature analysis objects for musical programming environments like pure data (Pd) are useful as both a source of control information and as a toolkit for informal experimentation. This paper introduces a set of Pd externals developed by the author for this type of research: specBrightness~, specFlatness~, specRolloff~, specFlux~, specCentroid~, zeroCrossing~, cepstrum~, mfcc~, bfcc~, and timbreID. The report below evaluates their performance in a percussion instrument classification problem. In many real-time scenarios, results are needed immediately in order to classify an incoming sound and produce a synchronous event that depends on classification. With this in mind, a constraint of reporting results within 30 milliseconds was imposed.

2. METHOD

The process of choosing a suitable set of instruments for training and testing was guided by three desires: diversity of material, diversity of spectrum, and relatively short decay. The chosen training set includes a low tom, wooden plank, Chinese cymbal, nipple gong, cabaça (shaker), metal bowl, bongo, small anvil, tambourine, thundersheet, conga, and wooden box. A single Shure SM57 cardioid microphone was used as the input source. 20 training examples of each instrument were recorded, but only 5 were used in actual training. The test sequence consisted of one strike of each instrument in the order given above. 10 runs were recorded at a tempo of roughly 108 bpm, followed by 3 additional runs at roughly 180 bpm. With a total of 13 runs through the 12 instruments, the complete test consists of classifying 156 unique attacks.

Attacks were detected using bonk~ [5], which introduces a delay of at least 6 ms after an attack peak. Analyses were

PdCon09, July 19-26, , 2009, Sao Paulo, SP, Brazil.

performed using 1024-point windows (23 ms at a sampling rate of 44.1 kHz), where the end of the window is chosen relative to the attack report from bonk~. With this configuration, the first analysis window ends roughly 6 ms after an attack peak (which is itself a few ms after the true onset of the sound), and begins roughly 17 ms before the attack peak. The time between an instrument onset and its attack peak varies unpredictably according to the instrument in question (not to mention its pitch and dynamic), making it difficult to choose a general strategy for determining the optimal placement of an analysis window.

For this reason, many runs of analyses were performed systematically with a delay between the attack report and the analysis snapshot. This delay setting will be referred to as the post-attack analysis time, or AT. The snapshot described above corresponds to an AT setting of 0 ms. With an AT setting of 10 ms, however, the analysis window would start only 7 ms before the attack peak (rather than 17), reducing the presence of any pre-onset resonance that is not related to the desired object of analysis. These values and their relationship to true onsets are unavoidably approximate, but the range of AT settings between 0 and 23.22 ms employed here are sufficient for discussing the effects of snapshot timing. The feature externals are designed to report analysis results as a list immediately upon a bang, with sample-accurate timing. This level of accuracy is not necessarily beneficial when using bonk~, which only reports attacks at the start of 64-sample DSP blocks. However, for analysis requests generated by metro or delay objects, such precise timing may be significant.

Two general analysis schemes were implemented: single- and multiple-frame. The former has the advantage of reducing latency as only one snapshot is taken. The total delay is thus the delay of bonk~ plus the AT setting. With multiple-frame analysis, 10 successive overlapping frames are analyzed, capturing the initial temporal evolution of the sound. The delay between each frame is 64 samples, or 1.4512 ms, so the total latency incurred is that of bonk~ plus AT plus 13.061 ms. The advantage, however, is a significant score improvement. In situations where classification results are not needed immediately, the analysis externals can be banged repeatedly with a metro to track the temporal timbre evolution of entire sound events.

All of the externals buffer audio and perform Hann windowing internally based on a user-specified window size *N*. When a bang is received, analysis results for the window beginning *N* samples earlier are output as a list. Any number of such lists can be packed together and manipulated in Pd before being sent as a feature vector to the timbreID classification external. Although knowledge of the analysis techniques described below will be helpful, the externals are designed to be very simple to use.

2.1 Features

This section provides a general description of the features studied and their implementation in Pd.

2.2 Spectral Brightness

Spectral Brightness is the ratio of the sum of magnitudes above a given boundary frequency *f(K)* to the sum of all magnitudes in a spectrum. Signals with a significant amount of high frequency content will have higher brightness.

$$B_f = \frac{\sum_{k=K}^{N/2} |X(k)|}{\sum_{k=0}^{N/2} |X(k)|} \tag{1}$$

Arbitrary values of *f(K)* can be set for specBrightness~, but the default is 1200 Hz, or K=28 for a 1024-point analysis window at a sampling rate of 44.1 kHz.

2.3 Spectral Flatness

Spectral Flatness is the ratio of the geometric mean of magnitude spectrum to the arithmetic mean of magnitude spectrum. A very noisy spectrum without clear shape (i.e. that of white noise) should have a high flatness value, where 1.0 is perfect flatness.

$$B_f = \frac{\sqrt[N/2]{\prod_{k=0}^{N/2} |X(k)|}}{\frac{1}{N/2} \sum_{k=0}^{N/2} |X(k)|} \tag{2}$$

2.4 Spectral Roll-off

Spectral Roll-off is the frequency of bin *K*, *f(K)*, below which a certain percentage of total spectral energy is concentrated. A typical amount is 85%, but arbitrary concentration values can be set for specRolloff~.

$$max\{f(K) : \sum_{k=0}^{K} |X(k)| \leq 0.85 \sum_{k=0}^{N/2} |X(k)|\} \tag{3}$$

2.5 Spectral Flux

Spectral Flux is sum of squared difference between two successive magnitude spectra. The time separation between successive frames can be specified in samples for specFlux~. 128 samples is the default value.

$$F = \sum_{k=0}^{N/2} \left[|X_i(k)| - |X_{i-1}(k)| \right]^2 \tag{4}$$

2.6 Spectral Centroid

Spectral Centroid is the center of mass of magnitude spectrum. It is computed as the ratio of the sum of spectral magnitude weighted by frequency to the sum of spectral magnitude.

$$C = \frac{\sum_{k=0}^{N/2} f(k) \, |X(k)|}{\sum_{k=0}^{N/2} |X(k)|} \tag{5}$$

where *f(k)* is the frequency associated with bin *k*. Therefore, specCentroid~ reports the frequency associated with spectral center of mass.

2.7 Zero Crossing

Zero crossing is a time-domain feature, and is simply the number of times the waveform crosses zero in the window. It is computed as

$$Z = \sum_{n=2}^{N} sign[x(n)] - sign[x(n-1)] \quad (6)$$

where *sign* is the signum function, which returns 1 when the argument is positive, -1 when it is negative, and 0 otherwise.

2.8 Cepstrum

Real cepstrum (x_{RC}) is defined in [6] as

$$x_{RC}(n) = \Re[\ IFT\{ln\,|X(k)|\}\ ] \quad (7)$$

where $X(k)$ is the frequency domain representation of a signal $x(n)$, and $\Re$ denotes the real portion of the inverse Fourier transform. The real cepstrum is sometimes equivalently defined as the real part of the *forward* Fourier transform of the logarithm of magnitude spectrum. The first 40 or so cepstral coefficients of a 1024-point window provide a compact representation of spectral envelope.

2.9 Mel Frequency Cepstrum

When cepstrum is weighted according to perceptual scales, it functions more effectively as a timbral feature descriptor. One such scale unit is the mel, proposed in [7]. A general formula for calculating mels is

$$Mel(f) = 2595 \times log_{10}(1 + \frac{f}{700}) \quad (8)$$

where f is frequency in Hz.

The process for computing Mel Frequency Cepstral Coefficients (MFCCs) differs from raw cepstrum computation considerably. It requires a bank of overlapping triangular bandpass filters evenly spaced on the mel scale, and the final transform is a discrete cosine transform (DCT) rather than a Fourier transform. MFCCs are defined mathematically as

$$MFCC_i = \sum_{k=0}^{N-1} X_k\ cos[i(k+\frac{1}{2})\frac{\pi}{N}];\quad i = 0, 1, \ldots, M-1 \quad (9)$$

where M is the number of desired cepstral coefficients, N is the number of filters, and X_k is the log power output of the kth filter. Mel scaling and smoothing significantly reduces the size of spectral envelope data and emphasizes lower frequency content.

As MFCCs are more complex than the low level features listed above, adjusting parameters for mfcc∼ can have more significant repercussions. For instance, the mel spacing between filters in the filterbank can be changed from the 100 mel default value. Reducing the spacing parameter results in the creation of more filters in the filterbank, which makes the feature vector longer and preserves more of the detail in the original spectrum. It also reduces the effect of mel weighting.

2.10 Bark Frequency Cepstrum

Implementation of Bark weighting in place of mel weighting is straightforward. The collection of frequencies used for filterbank construction are simply generated based on a different equation. Equation (10), taken from [8], will be used here, where f is frequency in Hz.

$$Bark = [26.81 \times \frac{f}{(1960 + f)}] - 0.53 \quad (10)$$

Similar to mfcc∼, bfcc∼ has a Bark spacing parameter for filterbank construction, which is half a Bark by default.

3. CLASSIFICATION

Classification can be performed by interfacing the outputs of feature extraction objects with timbreID, an external that accumulates training example feature vectors as templates and compares incoming vectors against the stored data. A choice of either Euclidean distance or Manhattan (taxicab) distance is computed between the incoming vector and all training templates. The template that is nearest in the feature space is taken to be the best match, and its template index is sent out of timbreID's left outlet.

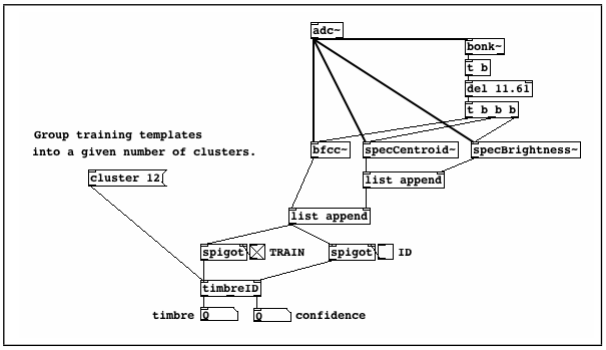

Figure 1: An example patch combining BFCCs, centroid, and brightness into a single feature vector sent to timbreID.

Templates can be grouped together according to instrument using the "cluster" message, which performs a hierarchical clustering algorithm. This process should automatically put timbrally similar training examples in the same cluster, so that when a matching template is found, its cluster index rather than its template index is output. In cases where automatic clustering fails or dissimilar templates need to be grouped together, a "manual_cluster" message is also available that allows any sequence of templates to be placed in the same cluster.

Once automatic or manual clustering has been performed, timbreID's right outlet will report a confidence measure between 0.0 and 1.0 reflecting the ratio of the nearest match to the next nearest match from another cluster in terms of distance from the input signal. This could be useful for ignoring classifications that have a confidence rating below a specified threshold.

In the event that some components of the feature vector are thought to be more important than others, weights can

be specifically assigned for distance computation with the "weights" message.

3.1 Multiple Frame Analysis

Keeping the analysis externals separate from the classification external provides flexibility that makes many analysis techniques possible. Multiple-channel and multiple-frame analyses are the most promising. Although the goal is to classify sounds as quickly as possible, the tests below show that accepting a small amount of additional latency in order to perform a series of analyses spaced 1.45 ms apart can increase accuracy. This technique captures the beginnings of temporally evolving timbre characteristics in the initial moments of a sound.

In Figure 2, a patch is shown that performs ten sequential Bark-frequency cepstrum analyses. The resulting feature vector is graphed in Figure 3, showing the slight change in BFCCs from one frame to the next.

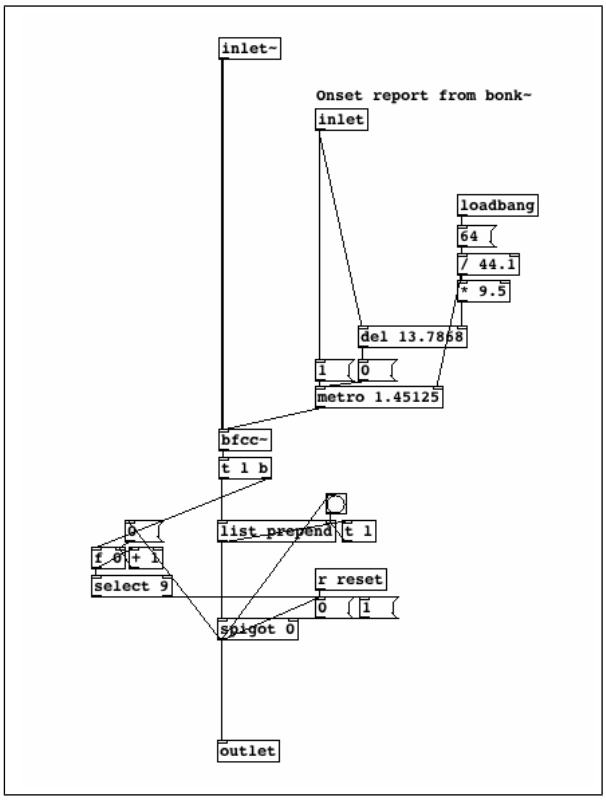

Figure 2: An example patch accumulating BFCC vectors in a list.

4. PERFORMANCE RESULTS

This section presents scores for the classification of 156 instrument onsets using single-frame analysis. The performance of each feature external is reported individually first, followed by scores generated using combined lists of multiple features. Because bfcc∼ has previously been shown to be the most accurate of the cepstrum-based externals, cepstrum∼ and mfcc∼ will not be tested here [1].

The tables below are formatted uniformly, showing the AT setting on the upper row and corresponding scores on the lower row. AT is adjusted by 1.4512 ms increments until 18.87 ms, which is followed by 23.22 ms. Scores are normalized to the 0.0 - 1.0 range, where 1.0 reflects a perfect score of 156 accurate classifications.

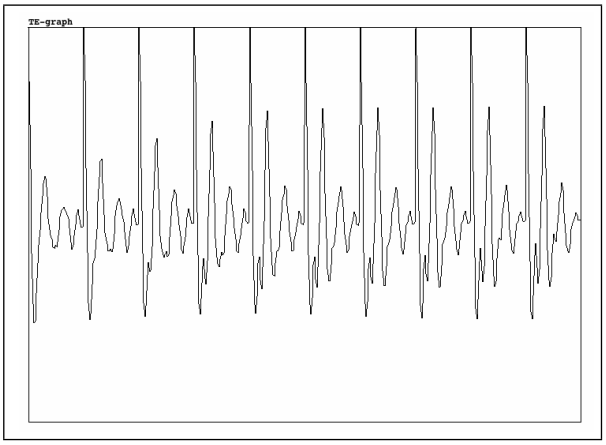

Figure 3: A graph showing sequential BFCC analyses.

4.1 Single Frame Performance Results

AT	0.00	1.45	2.90	4.35	5.81
Score	0.37	0.40	0.44	0.41	0.51

AT	7.26	8.71	10.16	11.61	13.06
Score	0.47	0.44	0.39	0.42	0.53

AT	14.51	15.96	17.42	18.87
Score	0.42	0.49	0.48	0.47

Table 1: single-frame specBrightness∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.28	0.27	0.29	0.37	0.36

AT	7.26	8.71	10.16	11.61	13.06
Score	0.45	0.38	0.42	0.38	0.39

AT	14.51	15.96	17.42	18.87
Score	0.40	0.45	0.50	0.53

Table 2: single-frame specFlatness∼

Bark-frequency cepstrum results in Table 7 stand far above the low level features in terms of score. With AT settings at 13.06 ms and above, there are three cases of a perfect score of 156 correct classifications.

The scores of individual low level features are not very encouraging; however, when all features are combined into a single vector, there is some slight improvement. The highest of the individual low level feature scores were those of specRolloff∼ and specCentroid∼ near 65%. The complete set of low level features generate scores as high as 75%.

Although the high scores of the single-frame BFCC analysis

AT	0.00	1.45	2.90	4.35	5.81
Score	0.49	0.53	0.46	0.49	0.48

AT	7.26	8.71	10.16	11.61	13.06
Score	0.53	0.58	0.58	0.56	0.64

AT	14.51	15.96	17.42	18.87
Score	0.55	0.51	0.56	0.56

Table 3: single-frame specRolloff∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.12	0.13	0.22	0.12	0.15

AT	7.26	8.71	10.16	11.61	13.06
Score	0.22	0.14	0.17	0.26	0.22

AT	14.51	15.96	17.42	18.87
Score	0.13	0.20	0.19	0.19

Table 4: single-frame specFlux∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.47	0.49	0.56	0.58	0.58

AT	7.26	8.71	10.16	11.61	13.06
Score	0.60	0.67	0.68	0.60	0.58

AT	14.51	15.96	17.42	18.87
Score	0.58	0.53	0.58	0.57

Table 5: single-frame specCentroid∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.08	0.08	0.08	0.09	0.08

AT	7.26	8.71	10.16	11.61	13.06
Score	0.14	0.17	0.15	0.19	0.36

AT	14.51	15.96	17.42	18.87
Score	0.50	0.58	0.63	0.58

Table 6: single-frame zeroCrossing∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.94	0.94	0.95	0.96	0.96

AT	7.26	8.71	10.16	11.61	13.06
Score	0.96	0.97	0.99	0.98	1.00

AT	14.51	15.96	17.42	18.87
Score	0.99	1.00	1.00	0.99

Table 7: single-frame bfcc∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.53	0.60	0.59	0.65	0.61

AT	7.26	8.71	10.16	11.61	13.06
Score	0.69	0.71	0.71	0.73	0.74

AT	14.51	15.96	17.42	18.87
Score	0.69	0.72	0.71	0.70

Table 8: single-frame combined low level features

AT	0.00	1.45	2.90	4.35	5.81
Score	0.52	0.58	0.57	0.63	0.69

AT	7.26	8.71	10.16	11.61	13.06
Score	0.65	0.69	0.78	0.76	0.75

AT	14.51	15.96	17.42	18.87
Score	0.71	0.71	0.74	0.70

Table 9: single-frame bfcc∼ + low level features

do not leave very much room for improvement, there remains the chance that extra features could actually produce lower scores. The results of concatenating the BFCC vector with the six low level features show that this is indeed the case.

4.2 Multiple Frame Performance Results

Scores given in Tables 10-18 below show that many of the individual low level features are much more effective in multiple-frame analysis. As with the single-frame case, concatenating all of them results in a slight score improvement (Table 17).

In the multiple-frame scenario, BFCCs remain the most effective single feature, this time generating many more perfect scores at earlier AT settings.

When concatenating the multiple-frame vector of BFCCs with those of the six low level features, the additional information is apparently helpful in correcting some of the slight inaccuracy seen at later AT settings for the pure BFCC feature. Although the complete vector is quite long (530-point) in comparison to the 47-point single-frame BFCC, the cost of a distance calculation is low enough to be very manageable in real time depending on the number of stored training templates. The tests here require 60 comparisons per classification request.

AT	0.00	1.45	2.90	4.35	5.81
Score	0.70	0.74	0.74	0.74	0.74

AT	7.26	8.71	10.16	11.61	13.06
Score	0.74	0.74	0.74	0.69	0.67

AT	14.51	15.96	17.42	18.87
Score	0.67	0.69	0.68	0.69

Table 10: multiple-frame specBrightness∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.63	0.65	0.60	0.65	0.60

AT	7.26	8.71	10.16	11.61	13.06
Score	0.60	0.61	0.58	0.56	0.58

AT	14.51	15.96	17.42	18.87
Score	0.56	0.54	0.55	0.55

Table 11: multiple-frame specFlatness∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.79	0.81	0.81	0.82	0.83

AT	7.26	8.71	10.16	11.61	13.06
Score	0.82	0.80	0.81	0.78	0.73

AT	14.51	15.96	17.42	18.87
Score	0.74	0.73	0.72	0.77

Table 12: multiple-frame specRolloff∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.29	0.29	0.33	0.32	0.33

AT	7.26	8.71	10.16	11.61	13.06
Score	0.35	0.37	0.41	0.38	0.40

AT	14.51	15.96	17.42	18.87
Score	0.34	0.33	0.38	0.37

Table 13: multiple-frame specFlux∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.83	0.83	0.79	0.80	0.81

AT	7.26	8.71	10.16	11.61	13.06
Score	0.79	0.77	0.81	0.79	0.81

AT	14.51	15.96	17.42	18.87
Score	0.78	0.78	0.75	0.71

Table 14: multiple-frame specCentroid∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.26	0.26	0.31	0.31	0.40

AT	7.26	8.71	10.16	11.61	13.06
Score	0.42	0.51	0.56	0.67	0.71

AT	14.51	15.96	17.42	18.87
Score	0.72	0.73	0.68	0.71

Table 15: multiple-frame zeroCrossing∼

AT	0.00	1.45	2.90	4.35	5.81
Score	1.00	1.00	1.00	1.00	1.00

AT	7.26	8.71	10.16	11.61	13.06
Score	1.00	1.00	1.00	0.97	1.00

AT	14.51	15.96	17.42	18.87
Score	0.99	0.99	0.99	0.99

Table 16: multiple-frame bfcc∼

AT	0.00	1.45	2.90	4.35	5.81
Score	0.84	0.87	0.85	0.86	0.84

AT	7.26	8.71	10.16	11.61	13.06
Score	0.83	0.80	0.81	0.83	0.82

AT	14.51	15.96	17.42	18.87
Score	0.85	0.84	0.87	0.87

Table 17: multiple-frame concatenated low level features

AT	0.00	1.45	2.90	4.35	5.81
Score	1.00	1.00	1.00	1.00	1.00

AT	7.26	8.71	10.16	11.61	13.06
Score	1.00	1.00	1.00	1.00	1.00

AT	14.51	15.96	17.42	18.87
Score	1.00	1.00	1.00	1.00

Table 18: multiple-frame bfcc∼ + low level features

5. CONCLUSIONS AND FUTURE WORK

While BFCCs are clearly the most useful feature, two of the low level features—centroid and roll-off—stand out as well. On the other hand, spectral flux produced particularly low scores, making its usefulness questionable. While none of the low level feature scores are individually useful for general applications, perhaps the scores reported here could aid in choosing appropriate weights to be used when combining features.

Capturing the earliest segment of temporal evolution with multiple-frame analysis was beneficial across the board, and generated perfect scores in the case of tests using bfcc~ and bfcc~ + all low level features. Perfect scores began at an AT setting of 0 ms, meaning the only latency involved is that of bonk~'s attack detection (around 6 ms) plus the 9 additional analysis frames at 1.45 ms each. This results in latency around 19 ms—low enough to be useful for immediate use in real-time applications.

At present, both the basic tests described above and early musical experiments by the author indicate that mfcc~ and bfcc~ are sufficiently robust to be useful in real-time applications. Source code for the entire set of externals is available at http://www.williambrent.com.

Future development will focus on the challenge of polyphonic classification—searching for a combination of training templates that best explains the content of the analysis window.

6. REFERENCES

[1] W. Brent. Perceptually based pitch scales in cepstral techniques percussive timbre identification. In *Proceedings of the International Computer Music Conference*, Montreal, Canada, 2009.

[2] J. Grey. *An Exploration of Musical Timbre Using Computer-based Techniques for Analysis, Synthesis and Perceptual Scaling*. PhD thesis, Stanford University, 1975.

[3] P. Iverson and C. Krumhansl. Isolating the dynamic attributes of musical timbre. *Journal of the Acoustical Society of America*, 94(5):2595–2603, 1993.

[4] O. Lartillot and P. Toiviainen. A matlab toolbox for musical feature extraction from audio. In *Proceedings of the 10th International Conference on Digital Audio Effects*, Bordeaux, France, 2007.

[5] M. Puckette, T. Apel, and D. Zicarelli. Real-time audio analysis tools for pd and msp. In *Proceedings of the International Computer Music Conference*, pages 109–112, 1998.

[6] T. Quatieri and J. Tribolet. Computation of the real cepstrum and minimum phase reconstruction. In *Programs for Digital Signal Processing*, pages 7.2–1–7.2–6, New York: IEEE Press, 1979.

[7] S. Stevens, J. Volkman, and E. Newman. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 8:185–190, 1937.

[8] H. Traunmüller. Analytical expressions for the tonotopic sensory scale. *Journal of the Acoustical Society of America*, 88(1):97–100, 1990.