# EFFICIENT ACOUSTIC FEATURE EXTRACTION FOR MUSIC INFORMATION RETRIEVAL USING PROGRAMMABLE GATE ARRAYS

**Erik M. Schmidt**
MET-lab, Drexel University
`eschmidt@drexel.edu`

**Kris West**
IMIRSEL, University of Illinois
`kris.west@gmail.com`

**Youngmoo E. Kim**
MET-lab, Drexel University
`ykim@drexel.edu`

## ABSTRACT

Many of the recent advances in music information retrieval from audio signals have been data-driven, i.e., resulting from the analysis of very large data sets. Widespread performance evaluations on common data sets, such as the annual MIREX events, have also been instrumental in advancing the field. These endeavors incur a large computational cost, and could potentially benefit greatly from more rapid calculation of acoustic features. Traditional, cluster-based solutions for large-scale feature extraction are expensive and space- and power-inefficient. Using the massively parallel architecture of the field programmable gate array (FPGA), it is possible to design an application specific chip rivaling the speed of a cluster for large-scale acoustic feature computation at lower cost. Recent advances in development tools, such as the Xilinx Blockset in Simulink, allow rapid prototyping, simulation, and implementation on actual hardware. Such devices also show potential for the implementation of MIR systems on embedded devices such as cell phones and PDAs where hardware acceleration would be an absolute necessity. We present a prototype library for acoustic feature calculation for implementation on Xilinx FPGA hardware. Furthermore, using a genre classification task we compare the performance of simulated hardware features to those computed using standard methods, demonstrating a nearly negligible drop in classification performance with the potential for large reductions in computation time.

## 1. INTRODUCTION

The extraction of appropriate acoustic features is the first step for nearly all audio-based music information retrieval applications. Many recent advances in MIR systems are the result of large-scale, data-driven analysis of audio samples. Such corpora may contain thousands (or even millions, in the case of some commercial databases) of audio files, and the accompanying analyses of these data sets demands vast computational resources. In seeking improved methods for music classification and understanding, researchers are constantly searching for more informative feature sets, which requires the ability to rapidly prototype and evaluate new features on very large databases. Additionally, performance evaluations, such as the annual MIREX events [1], of multiple approaches to specific application tasks on common data sets have proven to be invaluable for advancing the state-of-the-art in MIR research. These evaluations, however, are increasingly difficult to administer, since both the number of participants and the size of the data sets continues to grow annually.

The most common solution to problems having such computational demands involves an investment in computing clusters, which are expensive and inefficient (in terms of both their utilization of hardware resources and energy consumed). Using the massively parallel architecture of the field programmable gate array (FPGA) it is possible to achieve parallel processing on a scale similar to that of a small cluster (for specific applications) on a single chip. Current tools such as the Xilinx System Generator (XSG) for DSP [1] enable rapid prototyping of DSP algorithms in the graphical language of Simulink with the ability to incorporate hardware in the modeling and design loop. Additionally, any algorithm built using the Simulink Xilinx Blockset can be easily compiled into Verilog or VHDL code and incorporated into a larger hardware system design. One possible implementation would be MIR on embedded, mobile devices, such as cell phones and PDAs, where the computation of acoustic features would not be possible using the onboard CPU. Such a hardware acceleration system could be designed both for the computation of acoustic features, and evaluation of the decision function of a pre-trained classifier.

We have developed an acoustic feature extraction library, implemented using XSG, that can be synthesized directly on supported FPGA hardware. The library supports the calculation of both Mel-Frequency Cepstral Coefficients (MFCC) [2] and common Statistical Spectrum Descriptors (SSDs). Here, we present preliminary classification results using fixed-point MFCC features calculated by the XSG (simulating an FPGA implementation). The accuracy of these features is verified through their use in a genre classification task on a medium-sized audio database, and we demonstrate that the resulting classification performance is comparable to that of a floating-point MATLAB and double-precision Java implementation of similar feature cal-

[1] Xilinx System Generator: `http://www.xilinx.com/ise/optional_prod/system_generator.htm`

culation algorithms.

Furthermore, we also present a process for migrating acoustic features designed using MATLAB to the Xilinx System Generator in Simulink. The ultimate goal of this endeavor is to develop a system in which features can be rapidly and easily prototyped within Simulink, synthesized to Verilog hardware description language (HDL), and embedded into a larger standalone FPGA-based system-on-chip design. Tools developed for such a platform could be easily shared between members of the MIR research community and could potentially allow even an inexperienced HDL programmer to take advantage of the performance gained from implementing feature extraction in hardware.

## 2. BACKGROUND

Early efforts at implementing acoustic feature computation in hardware required systems built entirely from scratch, which is a significant and time-consuming endeavor when working directly with low-level hardware descriptor languages (HDLs). Prior work has almost exclusively targeted MFCC feature calculation for automatic speech recognition. For example, [3] presents an optimized algorithm for efficient computation of MFCCs using an FPGA implementation, while [4] focuses on implementing only the FFT sub-calculation in hardware for eventual use in MFCC computation. In the direction of easing the dependence on hardware arithmetic units, [5] proposes modifying the MFCC algorithm from a triangular filterbank to a mel-spaced rectangular filterbank, and their results demonstrate only a minimal decrease in classification accuracy. Other work has focused on full hardware system integration for speech recognition. In [6] an on-chip, retrainable hardware speech recognition system is presented using MFCC features and Hidden Markov Models (HMMs) for statistical pattern recognition.

The annual Music Information Retrieval Evaluation eXchange (MIREX) tasks, initiated in 2005, have become a core component of the field of MIR in terms of advancing and disseminating the latest research and results. With the number of tasks, participants, and data sets increasing annually, the evaluations have become exponentially more difficult to administer. The International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL), the organizers of MIREX, has traditionally gone above and beyond in order to accommodate a wide range of implementation platforms and architectures and retains a range of machines, including an ever-expanding cluster, for this purpose, resulting in additional complexity. This configuration, while offering a great deal of flexibility, limits their ability to take advantage of parallel processing implementations, which are still highly platform-specific. A 72-hour runtime cap is enforced for all submissions, but even so, recent evaluations have required up-to 1000 person-hours of effort. A significant speedup in feature computation could greatly reduce the runtime requirements for MIREX.

## 3. HARDWARE IMPLEMENTATION OF FEATURES

The Xilinx System Generator (XSG) tools in Simulink enable the prototyping of complex signal processing algorithms for hardware implementation in a relatively straightforward manner. For our initial implementation, we limited ourselves to analysis windows with a length of 512 samples and 50% overlap, 40 triangular mel-filters, and 20 DCT coefficients. These parameters were chosen because of their wide application in audio and music processing algorithms. Other research has shown that the number of filters and DCT coefficients can be greatly reduced while still maintaining adequate performance [7].

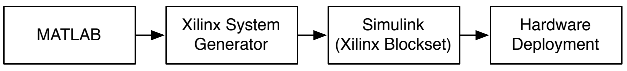

**Figure 1**. Implementation flowchart for audio feature extraction algorithms.

The first processing stage for our audio features requires a DFT calculation, which is performed using the pipelined Xilinx core FFT v5.0, producing real-time serial FFT data. Next, using two multipliers, we square the results of the real and imaginary parts and add them together to obtain the energy spectral density (ESD), which is quantized to 32-bits. In this case, ESD is preferred to the magnitude FFT because the square root function necessary to obtain magnitude is not easily implemented in hardware. XSG includes Coordinate Rotation Digital Computer (CORDIC) algorithms which can compute square roots as well as trigonometric, logarithmic, and division functions using only addition, subtraction, bitshift, and table lookup, but in a practical design these algorithms tend to take up large areas of the FPGA fabric and often incur large delays. In general, we avoided the use of these functions whenever possible.

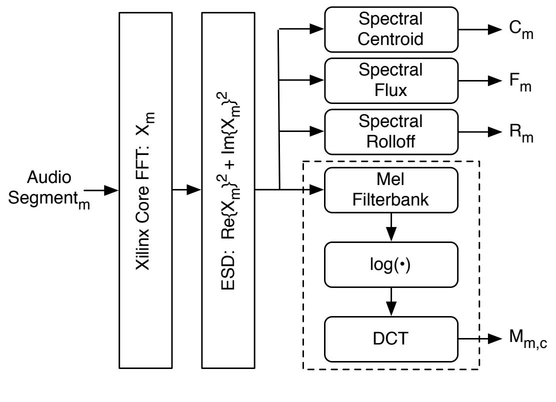

**Figure 2**. Hardware implementation of audio feature computation.

## 3.1 Mel-Frequency Cepstral Coefficients

### *3.1.1 Algorithm*

Mel-frequency cepstral coefficients (MFCCs) are among the most widely used acoustic features in speech and audio processing. MFCCs are essentially a low-dimensional representation of the spectrum warped according to the mel-scale, which reflects the nonlinear frequency sensitivity of the human auditory system [2]. In our implementation, MFCCs are defined as

$$M_{m,c} = \sum_{b=1}^{40} \hat{X}_{m,b} \cos\left[c\left(b-\frac{1}{2}\right)\frac{\pi}{40}\right], c = 1,2,...,20 \tag{1}$$

$$\hat{X}_{m,b} = \log\left(f_b[k]\left|\sum_{n=0}^{N-1} x_m[n]e^{-j\frac{2\pi nk}{N}}\right|\right), \tag{2}$$

where $m$ represents the current frame. Normally, MFCCs are implemented over short-time segments, and accordingly our implementation divides the audio into overlapping segments and applies a Hanning window function to reduce edge effects. The Discrete Fourier Transform (DFT) of each short-time segment is computed using the FFT algorithm. The magnitude of the frequency components is determined and $f_b[k]$, the mel-spaced triangular filters (Figure 3), are applied via multiplication in the frequency domain. Continuing with the cepstrum calculation, the $\log$ of the mel-filtered energies is calculated ($\hat{X}_{m,b}$), which to some extent, serves to deconvolve the audio by transforming multiplications in the frequency-domain (and thus, convolutions in the time-domain) into additions. As a final step, the inverse DFT is applied to $\hat{X}_{m,b}$ using the DCT (sine components are not needed since the input is guaranteed to be real and even). This step is also used to reduce the dimensionality of the data to the desired quantity, and it has been shown that the DCT has the additional effect of decorrelating the vector of feature components [8].

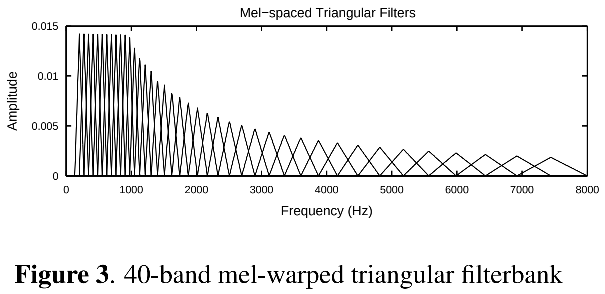

**Figure 3**. 40-band mel-warped triangular filterbank

### *3.1.2 FPGA Implementation*

Calculation of the MFCC features consists of applying the mel-filterbank to the spectrum, taking the $\log$, and computing the DCT. Applying the mel-filterbank requires 40 read-only memory (ROM) elements, 40 multipliers, and 40 accumulators. A control register is placed on the output which is triggered by FFT completion to only allow the filterbank output to change once for every frame. To minimize the size of the ROM elements and the number of multiplies, the mel-filter coefficients are restricted to 16-bits. This stage is the most resource intensive part of the design due to the number of multipliers required.

Once the data is filtered, it is again serialized in order to compute the $\log$. This requires a wait of 512 samples until the next FFT frame is supplied, therefore ample time is available to serialize the 40 filter band values. Using a single CORDIC $\log$, all 40 values are computed and subsequently quantized to 32-bits.

The final step involves computation of the DCT, where the DCT coefficients are stored in 20 ROM units and have been quantized to 16-bit resolution. In addition, this step requires 20 multipliers and 20 accumulators. The output is triggered using a simple control register such that the output values change only once every 512 values. Since no further processing is required, this data is decimated by a factor of 512 and returned as the final output.

## 3.2 Statistical Spectrum Descriptors

In music and audio processing, Statistical Spectrum Descriptors (SSDs) are often related to timbral texture [9]. For each spectral shape function, we begin by dividing the data into short-overlapping segments, applying a Hanning window, and computing the magnitude DFT.

### *3.2.1 Spectral Centroid*

Spectral centroid is defined as the weighted-average (center of mass) of the spectrum,

$$C_m = \frac{\sum_{k=0}^{K-1} F[k]|X_m[k]|}{\sum_{k=0}^{K-1} |X_m[k]|}, \tag{3}$$

where $X_m[k]$ is the DFT of short-time segment $m$, and $F[k]$ is a vector of frequencies corresponding to the bins of the magnitude spectrum.

Computation of the spectral centroid requires a multiplication, two accumulators, and a division. Here the spectrum is summed with one accumulator and the spectrum, multiplied by the respective spectral bin values, is summed by the other accumulator. After passing a control register to ensure only one value is returned for each frame, the result is divided by the CORDIC divider provided by XSG.

### *3.2.2 Spectral "Flux"*

Spectral flux is defined as the Euclidean distance between successive spectral frames. We compute the square of this feature, which is defined as follows:

$$F_m = \sum_{k=0}^{K-1} (|X_m[k]| - |X_{m-1}[k]|)^2. \tag{4}$$

Again, $X_m[k]$ is the discrete spectrum of the current analysis frame $m$ and $X_{m-1}[k]$ is the spectrum of the previous frame.

Spectral flux is the simplest of all of the spectral shape features to compute. The hardware consists of a 512 sample delay block to maintain a copy of the previous frame, an adder, a multiplier, and an accumulator. An important note is that we are not computing the square root, but simply the sum of the squares as to avoid the use of additional CORDIC functions.

### *3.2.3 Spectral Rolloff*

Spectral rolloff is defined as the frequency beneath which a given proportion of the total spectral energy lies, typically 85%:

$$R_m = \frac{f_s}{K} r_m = \frac{f_s}{K} \left( \arg_{r_m} \sum_{k=0}^{r_m} |X_m[k]| = 0.85 \sum_{k=0}^{K-1} |X_m[k]| \right). \tag{5}$$

Here $|X_m[k]|$ is the magnitude of the $k$-th frequency sample of the current frame and $r_m$ is the frequency sample number that produces the desired 85% rolloff.

The core of the spectral rolloff implementation consists of two accumulators. The first accumulator sums the spectrum to obtain the total energy and multiplies it by 0.85, where as the second sums the delayed spectrum until the total energy reaches 85%. Once this value is obtained, the frequency value of the correspond spectral bin is returned. Of these features, spectral rolloff is probably the most robust to quantization effects as it is returning values of an already discretized function.

## 4. HARDWARE PERFORMANCE AND USAGE

The initial hardware target for this project is Digilent's Virtex-II Pro Development System. The Xilinx Virtex-II FPGA on the board (XC2VP30) contains 13,969 slices, 136 18-bit multipliers, 2,448Kb of block RAM, and two PowerPC Processors. While the available amount of FPGA fabric is highly constrained, at an academic discounted cost of $299.00 USD, the board is an attractive target, and its widespread adoption in education creates greater opportunities for algorithm experimentation and deployment.

Implementing only the ESD algorithm on this chip requires 299 slices, 8 18-bit multipliers, and no block RAM. Considering the total size and resource restrictions the most limiting factor is the number of multipliers required. Using all 136 multipliers and 5,083 slices we could compute the ESD for up to 17 analysis frames in parallel. For a single FFT implementation, the first frame requires 1123 clock cycles to compute and subsequent frames require an additional 512 cycles. With 17 in parallel, the first set of frames will still need 1123 clock cycles, although we will now output 17 frames at a time. Assuming the fabric is clocked at 80MHz, a conservative clock speed, a breakdown of performance is shown in Table 1.

We compared the hardware performance using a single FFT unit on an FPGA to that of the M2K toolkit [10] and MATLAB on a set of 600 audio clips, each 30 seconds in duration (the data set used in the classification task is detailed in the next section). The M2K and MATLAB features were calculated using a single processor core of a 2.4 GHz Intel Core 2 CPU. Table 2 reveals the computation times for each feature set, averaged across all 600 clips, and we observe that there is more than an order of magnitude difference between the hardware and software implementations. While MATLAB and M2K each produce results in just under a half second for each 30 second clip, the hardware requires only around 33ms. Additionally, it can be seen from Table 3 that as the number of FFT units available on the FPGA increases, the hardware can achieve sub 1ms computation times.

| FFTs | First frame ($\mu$s) | Subsequent frames ($\mu$s) | Three secs (ms) | Thirty secs (ms) |
|---|---|---|---|---|
| 1 | 14.04 | 6.400 | 3.316 | 33.08 |
| 17 | 14.04 | 0.376 | 0.206 | 1.953 |

**Table 1**. Performance of spectrogram calculation on simulated hardware.

| Toolkit | Computation Time (s) |
|---|---|
| M2K | 0.483 |
| MATLAB | 0.364 |
| FPGA | 0.033 |

**Table 2**. Comparison of feature computation times between software and hardware implementations.

In its current form, implementing the full system-on-chip (including MFCC and SSD calculations) with a single FFT unit requires 15,542 slices, 145 18-bit multipliers, and 1,080 Kb of block RAM, which exceeds the capacity (in terms of slices and multipliers) of the hardware target. With some additional optimization, perhaps trading off some parallelism to reduce hardware resource utilization, we believe the full system could be implemented on the targeted Virtex-II VC2VP30-based system. Other products in the Virtex-II FPGA family provide additional hardware resources, which offer the possibility of combining multiple feature computation engines on a single chip. For example, the Virtex-II XC2VP100 provides sufficient slices and multipliers to easily accommodate 3 feature computation engines. Additionally, certain development boards contain multiple FPGA chips, adding further parallelization opportunities. With two FPGAs, we could potentially accommodate 6-8 computation engines on a single system board.

The current design requires 2048 clock cycles to produce the first output for the spectral shape features and 2560 for MFCCs. As with the ESD, each additional frame takes 512 cycles and compute time decreases proportionally with the addition of each parallel computational engine. More specific timing results for a single computation engine (again clocked at a conservative 80 MHz) are provided in Table 3, as well as theoretical performance numbers if six parallel feature computation engines could be implemented on a single system. Although such development hardware is currently quite expensive (approximately $10K, on the order of a small cluster), the performance is potentially faster by more than an order of magnitude.

| Feature | Comp. Engines | First frame ($\mu$s) | Three secs (ms) | Thirty secs (ms) |
|---|---|---|---|---|
| S. Shape | 1 | 25.60 | 3.328 | 33.09 |
| S. Shape | 6 | 25.60 | 0.576 | 5.536 |
| MFCC | 1 | 32.00 | 3.334 | 33.10 |
| MFCC | 6 | 32.00 | 0.582 | 5.542 |

**Table 3**. Performance of feature extraction on simulated hardware.

## 5. CLASSIFICATION EXPERIMENT

In order to confirm the efficacy of the hardware extracted features, we have conducted an evaluation of genre classification systems based on the features produced and compared its performance to that of classifiers based on the same features computed in MATLAB and the M2K toolkit [10]. We conducted two experiments using a collection of 600 tracks drawn from the Magnatune collection of Creative Commons licensed music [11], divided into six genres. An overview of the collection is given in table 4.

| Genre | Number of tracks |
|---|---|
| Ambient | 100 tracks |
| Classical | 100 tracks |
| Electronic | 100 tracks |
| Ethnic | 100 tracks |
| Jazz and Blues | 100 tracks |
| Rock | 100 tracks |
| Total | 600 tracks |

**Table 4**. Composition of dataset for genre classification task.

Pampalk [12] identifies the potential for the over-fitting of the characteristics of a particular artist to inflate accuracy scores in the evaluation of audio content-based genre classification systems, particularly when evaluating systems on small collections. Hence, we have conducted both artist-filtered and conventional cross-validated classification experiments based on 5-fold random 80:20 splits and 5-fold stratified cross-validation, respectively.

### 5.1 Pre-processing of Features

The feature extractors yield a very large number of feature vectors for each track (based on 23 ms windows with 50% overlap), and in order to effectively and efficiently classify tracks, the features must be summarised to produce a smaller number of more informative vectors. One approach that has been effectively used by many authors [12–15] is to summarise the distribution of feature frames over the track. This may be performed by, for example, estimating the parameters of a single Gaussian distribution or a mixture of Gaussians. However, [16] and [12] provide experimental evidence that the performance of techniques based on mixtures of Gaussian distributions are at best equal to that of single Gaussian based approaches, making their extreme additional computational cost impossible to justify. This lack of additional discriminative power for the use of mixture distributions is at odds with research in many other audio indexing problems [16]. Hence, in our evaluation the feature stream is summarised as a flattened single Gaussian distribution (mean vector and and flattened upper triangular covariance matrix).

### 5.2 Classification Algorithms

The classification algorithms tested were drawn from the M2K toolkit [10] and Weka [17] and include: Fisher's Criterion Linear Discriminant Analysis (LDA) [18], Classification and Regression Trees (CART) [19], a first-order linear Support Vector Machine (based on John C. Platt's Sequential Minimal Optimisation, SMO, algorithm [20]) and the J48 decision tree algorithm [17].

### 5.3 Classification Results

The results of the artist-filtered and unfiltered classification experiments are given in Tables 5 and 6, respectively. For each classification method, the highest-performing feature set is highlighted in bold.

| Classifier<br>Feature set | CART | J48 | LDA | Linear SMO |
|---|---|---|---|---|
| M2K | **36.43%** | **36.79%** | 35.70% | 45.36% |
| Matlab | 34.24% | 35.15% | 37.16% | 46.63% |
| FPGA | 34.24% | 34.97% | **37.89%** | **49.00%** |

**Table 5**. Artist-filtered classification results.

| Classifier<br>Feature set | CART | J48 | LDA | Linear SMO |
|---|---|---|---|---|
| M2K | 41.67% | **44.50%** | 41.17% | **59.17%** |
| Matlab | 38.83% | 40.83% | 40.67% | 56.67% |
| FPGA | **42.67%** | 39.83% | **41.33%** | 57.50% |

**Table 6**. Cross-validated classification results.

## 6. DISCUSSION AND FUTURE WORK

The results above demonstrate that the implementation of acoustic feature computation in hardware can potentially reduce computation times by orders of magnitude, accompanied, at worst, by a nearly negligible decrease in classification accuracy. This initial implementation demonstrates a potential pathway for migrating MATLAB feature extraction code to the Xilinx Blockset in Simulink and ultimately to hardware. The current implementation, however, does not allow for easy deployment on FPGA hardware and additional challenges lie in integrating FPGA-based feature computation in a full MIR evaluation system.

Our next step is to create a custom platform for full hardware deployment in a standalone system. In this system, based on the Virtex-II Pro development board, the local computer will communicate with the FPGA board via gigabit ethernet. The onboard hardware PowerPC cores will ease development for the standalone platform by allowing us to write C code to manage the communication link with a host PC and the data flow in and out of the feature extraction logic.

Such a custom FPGA platform would allow algorithms to be quickly designed and tested in Simulink, and then compiled into Verilog code to be synthesized into the larger project. Given a full deployment system, it will be possible to run the system at much higher clock speeds than in the Simulink simulation and hardware-in-the-loop co-simulation. The fully deployed system will ultimately be a massively parallel system where multiple analysis windows are processed simultaneously.

## 7. ACKNOWLEDGEMENTS

The authors would like to thank the International Music Information Retrieval Systems Evaluation Lab (IMIRSEL) at the University of Illinois at Urbana Champaign for their help in implementing the experiments reported. This work is partially supported by National Science Foundation grant IIS-0644151.

## 8. REFERENCES

[1] J.S. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[2] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-28, No. 4:357–366, August 1980.

[3] J. C. Wang, J. F. Wang, and Y. S. Weng. Chip design of MFCC extraction for speech recognition. *Integration*, 32(1-2):111–131, Jan 2002.

[4] M. Nilsson and K. K. Paliwal. Speaker verification in software and hardware. *Microelectronic Engineering Research Conference*, 2001.

[5] W. Han, C. Chan, C. Choy, and K. Pun. An efficient MFCC extraction method in speech recognition. *Proceedings 2006 IEEE International Symposium on Circuits and Systems (ISCAS)*, page 4, Jan 2006.

[6] S. Nedevschi, R. Patra, and E. Brewer. Hardware speech recognition for user interfaces in low cost, low power devices. *Design Automation Conference, 2005. Proceedings. 42nd*, pages 684 – 689, May 2005.

[7] S. Sigurdsson, K. B. Petersen, and T. Lehn-Schiøler. Mel frequency cepstral coefficients: An evaluation of robustness of MP3 encoded music. In *Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR)*, 2006.

[8] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the First International Symposium on Music Information Retrieval (ISMIR)*, October 2000.

[9] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE Transactions on*, 10(5):293–302, 2002.

[10] J. S. Downie, A. F. Ehmann, and D. Tcheng. Music-to-knowledge (M2K): a prototyping and evaluation environment for music information retrieval research. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 676–676, New York, NY, USA, 2005. ACM.

[11] J. Buckman. Magnatune: Mp3 music and music licensing, April 2006.

[12] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Johannes Kepler University, Linz, March 2006.

[13] K. West. *Novel techniques for Audio Music Classification and Search*. PhD thesis, School of Computing Sciences, University of East Anglia, Norwich, United Kingdom, September 2008.

[14] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, August 2001.

[15] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of ISMIR 2005 Sixth International Conference on Music Information Retrieval*, 2005.

[16] J.-J Aucouturier. *Ten Experiments on the Modeling of Polyphonic Timbre*. PhD thesis, University of Paris 6, France, June 2006.

[17] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. *ICONIP/ANZIIS/ANNES*, pages 192–196, 1999.

[18] K. West and S. Cox. Features and classifiers for the automatic classification of musical audio signals. In *Proceedings of ISMIR 2004 Fifth International Conference on Music Information Retrieval*, 2004.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks/Cole Advanced books and Software, 1984.

[20] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods: Support Vector Learning*, 1999.