Unsupervised Classification of Music Signals:
Strategies Using Timbre and Rhythm

Zachary W. Bond

Submitted to the Faculty of Virginia Polytechnic Institute and State University in partial fulfillment of the requirements of the

Master of Science

degree in

Computer Engineering

Lynn Abbott
Louis Beex
Thomas Martin

November 15, 2006
Blacksburg, Virginia

Keywords: Music, classification, clustering, rhythm, timbre

Unsupervised Classification of Music Signals:
Strategies Using Timbre and Rhythm

Zachary W. Bond

Abstract

This thesis describes the ideal properties of an adaptable music classification system based on unsupervised machine learning, and argues that such a system should be based on the fundamental musical properties of timbre, rhythm, melody and harmony. The first two properties and the signal features associated with them are then explored in more depth. In the area of timbre, the relationship between musical style and commonly-extracted signal features within a broad range of piano music is explored, in an effort to identify features which are consistent among all piano music but different for other instruments. The effect of lossy compression on these same timbre features is also investigated. In the area of rhythm, a new tempo tracking tool is provided which produces a series of histograms containing beat and sub-beat information throughout the course of a musical recording. These histograms are then shown to be useful in the analysis of synthesized rhythms and real music. Additionally, a novel method based on the Expectation-Maximization algorithm is used to extract features for classification from the histograms.

Table of Contents

Figures and Tables

Figures:

Tables:

Chapter 1: Introduction

Between the advent of audio recordings in 1877 [46] and the emergence of Napster in 1999, most music collections were based on discrete storage units, whether wax cylinders, vinyl records, cassettes, or compact discs. In the 21st century, the confluence of high-quality audio compression, the growing storage capacity of hard drives, and the ubiquity of high-speed Internet have made it possible to transform large audio collections into digital archives on a home computer. The result is a need for tools to explore and organize these collections.

The purpose of this thesis is two-fold: At a high level, I introduce a strategy for audio classification using only the audio signal itself and some basic principles of music. This represents an achievable long-term goal. At a lower level, I explore two such musical properties, rhythm and timbre, examining the related features which may be extracted from audio signals. My approach is organized as follows: Because music classification draws from music theory, signal analysis, and pattern recognition, I provide background material in Chapter 2, including a discussion of the properties of music and a description of features drawn from previous music classification research. Such features form the foundation of the long-term classification goal. Also discussed are the unique issues facing potential classifiers. All this is used to argue that supervised classification based on pre-defined genres, the goal of most previous research, is inadequate for a high-quality audio classification and organization system, and that an adaptable, unsupervised approach is needed.

The next two chapters focus on specific musical properties and the music features associated with them, drawing all test cases from the realm of instrumental classical music. Chapter 3 explores the complex world of timbre, which has been heavily used in previous classification efforts. I test the consistency of a number of audio features commonly associated with timbre across a wide range of piano music, in an effort to learn which of the timbre features are independent of musical style. I also check to see which of the features remain constant even after MPEG Layer-3 (MP3) compression, as such compression results in a substantial loss of raw information, but little loss of human-perceived musical information. Chapter 4 is concerned with rhythm, which has also been well-researched because of its practical use to performers, mixers and disc jockeys. I

focus on charts known as beat histograms, which provide a description of the repetitions present in an audio signal. I present an alternative method of generating beat histograms, which allows for segmentation of audio by the rhythms contained within, and a novel strategy for extracting features from the histogram that are useful for classification.

Chapter 2: Background

2.1 Introduction

The success of music download services such as iTunes and eMusic, along with the popularity of file-sharing services like Limewire and Kazaa, highlights the significance of digital music in contemporary culture. The plethora of available audio, however, presents a significant challenge for consumers trying to navigate this musical maze. The research area of Musical Information Retrieval (MIR) has come into its own during the past decade as the need for reliable tools for finding and organizing collections of music has grown. MIR can be loosely divided into three basic research areas: music identification, music recommendation, and music classification.

Music identification is concerned with such goals as identification of songs irrespective of compression methods used and the identification of lead singer regardless of the song. Tools to identify audio include, at the simplest level, Compact Disc DataBase (CDDB) systems such as those by Gracenote [25] and freeDB [26]. A discussion of audio fingerprinting, which can be used to identify particular songs, is in [1]; a system for identification of songs by hummed tune is described in [2]. Readers can experiment with the latter and with some related tools on the Internet [27]. Singer identification has been attempted without much success in [19], but the results show promise.

Music recommendation is quite popular currently, with a wide array of approaches used to recommend new songs based on songs or artists already known to the user. The Music Genome Project [28] uses data determined by human listeners, whereas MusicIP Mixer and Playground [29] use features gathered automatically from audio files. The MusicIP system is proprietary, and thus difficult to judge; the Music Genome Project does not seem to exploit the full potential of its claimed database of music properties. Businesses such as Amazon and eMusic rely on the shopping history of other customers to recommend new titles.

Music classification is the subject of this thesis. It is directly related to music recommendation, but with the broader goal of finding trends and common groupings within a large music collection. Ideally, music classification would use only features

automatically generated from a user's audio files, so no information beyond the song collection itself is needed. However, before 2002 most research focused on symbolic representations of music, such as those used in Music Instrument Digital Interface (MIDI) [30] files. A recent overview of music classification from both perspectives can be found in [3]. Music classification is primarily useful for companies and individuals with large collections of music files, and provides the high-level organization of a collection. Music recommendation systems, by contrast, are best for very narrowly-focused exploration (such as playlist building), or for suggesting new music not currently in a collection. As the digital libraries of individuals continue to grow, the need for music classification tools will follow suit.

2.2 Defining Genres

Pieces of music are typically classified by experts and enthusiasts according to their genre, but this term tends to be vague and subjective. A quick look at the genre list at the music information database Allmusic [31] highlights the basic problem: There are many genres that are identified by country of origin ("British Rock"), that are associated with time periods ("Baroque"), or are distinguished by delivery mechanisms ("AM Rock") and marketing demographics ("Adult Contemporary"). It is true that each of these genres includes works that share particular sonic aspects; the problem is that the name does not describe those aspects, and that there are many pieces of music which fit in the genre but do not share a similar sound. There is also the problem of genre overlap, as it would be possible for a British band with music targeted to adults to be played on AM radio (although it is unlikely the music in question would also be from the Baroque era).

Another significant problem with this kind of pan-musical genre classification is its relative uselessness to individual non-experts, who would be better served by a system which adapts its classification to a particular collection. For example, a person who has a handful of tracks of electronic music will not find it useful for each track to be categorized into distinct sub-genres like "house" and "techno" whose differences he likely would not notice.

An overview of these and related issues in genre classification including the tradeoffs of different strategies can be found in [20]. For users with large collections, I believe an adaptive classification approach that generates genre hierarchies of a collection based on features derived automatically from audio files is the best way to tackle the problem. Before describing such a system, however, it is necessary to discuss some of the properties of music that need extraction, and some of the previous efforts at classification systems along with their drawbacks.

2.3 Basic Music Signal Features

At the highest level, music is considered to have four key properties: The *melody*, or sequence of pitches; the *harmony*, or the combinations of pitches; the *rhythm*, or organization of sounds in time; and the *timbre* or tone color, which is the property that gives each instrument or combination of instruments its distinctive sound [4]. Classification of music would ideally proceed based on these four properties. If music is represented in a symbolic form such as a MIDI file, these properties are quite straightforward to detect. For example, McKay in [5] offers a very thorough list of potential features, including interval (the space between simultaneous pitches) types and the relative occurrence of high and low pitched notes.

Generating music by electronic means from a symbolic form has a long history, but the inverse operation of generating symbolic representations from a performance is much more challenging for computers. One reason is that the way in which humans decode and comprehend music is not clearly understood, even if many aspects of psychoacoustics are. Some textbooks which discuss sound perception with an emphasis on music are [6] and [7]. Nonetheless, some work in related fields such as speech recognition has had applications in the realm of music classification. A brief overview of useful features in each category follows. A very thorough list of potential features for signals can also be found in [16].

Rhythm

The most developed area of music analysis is in the realm of rhythm, where DJs and others have a very practical need for systems that determine the tempo of popular music. Commonly, tempo extraction is performed using the low-frequency pulses as in [8], but such strategies are most useful for popular music genres that emphasize a heavy bass presence. More information can be found through the use of the signal autocorrelation to find self-similarities within the signal and therefore uncover rhythmic structure. Graphical representations of these similarities, called beat histograms, are discussed later in this thesis as well as in [14]. This approach can tackle tempo extraction as well as the more challenging problem of rhythm analysis on a small time scale (double versus triple meter, for example) and on a large time scale (different movements of a symphony, for example). The distribution of peaks in the histogram offers clues for the first problem, while segmentation helps handle the second.

Timbre

Timbre is the property which is second-most developed because an understanding of timbre is needed for synthesizers, and because timbre features have proved useful in other fields. Speech recognition, for example, relies on timbre to recognize vocal sounds regardless of the particular voice that is speaking. The Mel Frequency Cepstral Coefficients (MFCCs), introduced in [10], are a set of features useful for both speech recognition and music classification. Timbre is so well-used in music classification systems largely because features that correlate well with texture are relatively easy to extract. For example, the zero-crossing rate of an audio signal is directly related to its noisiness as perceived by a human, so it has been used to classify bass and snare drum sounds [23]. Other low-level features, such as the energy level of a sound and its decay over time also relate well to a human's perception of timbre, and these have been used in the classification of isolated tones [33]. Both these and the MFCCs are discussed in more detail later.

Melody and Harmony

Features based on melody can include the frequency of occurrence of particular pitches, and whether the pitches occurring in the music are associated with a particular

set of related pitches (tonality). Unfortunately, pitch identification by itself is a relatively challenging problem. The human brain detects the fundamental pitch of a tone, but in reality what we perceive as a single tone is actually a group of overtones at frequencies that are integer multiples of the fundamental. Sometimes, the fundamental is not even present in the sound. A promising but imperfect effort to transcribe synthesized harpsichord and commercially-recorded piano melodies into MIDI format can be found in [12]. This research and similar work is not yet directly useful for classification purposes due to limited accuracy in general and poor performance in realistic situations. Reliable pitch extraction must precede any reasonable features based on harmonic consonance or dissonance, so classification based on harmony is even farther out on the horizon. Features related to melody and harmony based on symbolic music representations in [5] might be useable in the signal domain once melody extraction is more dependable. In the MIDI domain, they have already been used with impressive success in a four-way composer classification problem [22].

It would be useful if a common framework could be used to extract all of the above features. Such a framework [14] has already been developed, and is called Marsyas (Music Analysis, Retrieval and SYnthesis for Audio Signals) after the Greek god of music. Its primary goal is to provide a dataflow model which encapsulates tools common to all kinds of music analysis and synthesis operations. In the realm of feature extraction, the initial data is an audio signal represented as a vector that undergoes transformations as it passes through a series of units called MarSystems, with the final output being a list of feature values. Marsyas handles a variety of input formats, though for this thesis it can be assumed that Marsyas automatically converts tracks to mono so as to handle a single waveform per file. The remaining chapters of this thesis use Marsyas extensively for extraction of features. For example, MarSystems to extract some of the timbre features described above already exist. Marsyas is primarily a Linux application distributed under the GNU Public License (GPL). It is available for download free from the Internet [37].

2.4 Previous Classification Work

Virtually all previous work in music classification has used supervised classification techniques. Neural networks and support vector machines are two of the classification approaches commonly adapted to music classification. In neural networks, weights are applied to threshold functions based on a pre-classified training set, so that the network learns relevant patterns in the data. In a Support Vector Machine (SVM), the feature space is projected into a higher-dimensional feature space with the hope of finding vectors that separate groups from one another. It is also possible to combine different classifiers to improve the overall result. The key element in all of these techniques is the use of a manually labeled training set, so these techniques are described as "supervised." A textbook that deals extensively with general supervised classification is [13]. When supervised strategies are used for music classification, the number of and types of genres are known at the start, and new genres cannot be added or removed during classification. However, accurately and consistently labeling the training set based on such a genre set is difficult for the reasons described in Section 2.2.

The earliest supervised music classification work began with Tzanetakis [14]. In this research, straightforward Bayesian and K-Nearest-Neighbor (KNN) classifiers were used. The former assumes a normal distribution for features of files within each genre, and uses the training set to estimate the parameters of a multivariate distribution; for the test set, a file is assigned to the genre for which its likelihood of membership is highest, based upon the distribution associated with the genre. In the KNN classifier, each file from the test set is mapped into a feature space, and it is assumed to belong to whichever genre is most common among its k nearest neighbors. The feature space used in [14] included timbre and rhythmic characteristics, and the data set contained popular and classical music; their results have been improved upon since then using other classification techniques, including neural networks [34] and SVMs [35]. However, these results are fundamentally difficult to compare due to the varying genre sets used in each paper. A comparison of results using a number of different feature sets is in [16]. In 2005, the Music Information Retrieval eXchange held a music classification contest [36], one part of which featured a ten-genre classification problem. Each participant used a 1005-file training set assigned to genres defined in advance and a 510-file test set. The

results reached a maximum accuracy of 77.75% for the 10-genre problem, which is representative of earlier efforts. The quality of classification is heavily dependent on the particular genre. For example, “Classical" was generally distinguished with 100% accuracy from other genres (see [15], for example), but genres with more nebulous labels, such as “Ambient” and “New Age,” were classified with far less accuracy. The list of genres (“Ambient,” “Blues,” “Classical,” “Electronic,” “Ethnic,” “Folk,” “Jazz,” “New Age,” “Punk,” and “Rock”) highlights the basic problem with supervised approaches: One can easily imagine genres that are not included in this classification but ought to be, and it is unclear exactly what some genres such as "Ethnic" refer to. Additionally, the disparity in breadth between genre labels (“Jazz” and “Classical” versus “Punk,” for example) is staggering.

An alternative to supervised clustering is unsupervised, which tries to group items without prior knowledge by simply noticing patterns and sorting appropriately. Many clustering methods are available to attack this problem. Some are statistically focused, and assume the presence of Gaussian distributions in the data. Perhaps the most popular of all methods is k-means, which iteratively discovers groups by finding the centroid of clusters and progressively improves the groups based on nearness to cluster centers. More visually-oriented methods, such as DBSCAN [18], try to classify groups whose members are densely packed, regardless of shape; this gives these methods an advantage in finding oddly shaped groups that are nevertheless obvious to a person. In any case, unsupervised clustering does not require a manually-classified training set, with the drawback that the clusters found by the computer will not have meaningful descriptions, and they will need to be given useful labels by the user.

To the author’s knowledge, unsupervised clustering has been applied to the problem of music classification from signals only twice. The first such paper [17] offers an exploration of a feature space organized using a hierarchical grouping procedure called Growing Hierarchical Self Organizing Maps (GHSOM). The authors create a feature space based on features derived from a creative psycho-acoustic model, and discuss the locations of files in their data set within the maps, although the resulting groups they uncover are differentiated primarily by beat strength. A related effort [21] uses a supervised process to select musical texture features based on their ability to

distinguish different groups of music by maximizing what the authors call ‘timbre distance,’ and then applies those results to create an Emergent Self-Organizing Map (ESOM). Their effort was primarily impeded by the fact that the groups used for supervised feature selection had a very wide within-group variance in sound with a lot of overlap between groups—the same basic problem with supervised classification, even if this approach was intended to lead to unsupervised classification. Given the variety of clustering algorithms, there is much room for their further use in the area of music classification; an attempt to cluster classical music in the timbre domain using DBSCAN is presented in the next chapter.

Based on the previous research, the of problem distinguishing very high-level genres such as “Rock” and “Jazz” using feature sets commonly available has been thoroughly studied. The time seems ripe to investigate how effective these same features are in an unsupervised situation. I have chosen to look at classifications within the non-vocal classical genre, which has only been touched upon in previous research. The diversity and enormous quantity of available recordings also makes it an ideal genre to explore in more depth. Before proceeding to experimentation in the realm of classical music, the next section discusses a bird's eye view of what would be an ideal classification system, presenting a long-term goal to be achieved.

2.5 An Ideal Classification System

An audio classification system, to be most useful to the end user, would have several key features, including adaptability to the user's collection and a hierarchical organization. Adaptability to the user means that the system should recognize common groupings within the user's collection of audio files, rather than rely on some external database of genre labels, and the hierarchical approach serves this need well. A user who has a dozen or so jazz tracks, for example, would probably be better off with all of them stuck into one genre (which he can label as "Jazz"), while a jazz enthusiast with thousands of files would need a much more detailed scheme. Clustering algorithms can be adapted to produce a hierarchical listing. DBSCAN, for example, has parameters indicating the smallest cluster size and the density required for groups to become clusters. By simply repeating the clustering process for each cluster, while increasing the density

value each time, sub-clusters can be found within larger groupings, until all clusters are of minimal size. Taken together this produces a full tree structure indicating cluster hierarchy. For efficiency, re-clustering would only occur when there have been significant changes to the audio collection. In general, newly added files would likely fall into already-discovered clusters and can be classified simply by putting them in the nearest group.

Of course, not everyone would wish to classify on exactly the same musical properties, so the user should specify whether to use rhythm, timbre, melody and harmony features at any given time. For example, suppose a DJ has successfully clustered all his electronic music away from other groups. Then he may no longer be interested in the particular types of electronic sounds in that genre, but only in the rhythmic characteristics. In that case, he would be able to classify the "electronic" genre exclusively based on rhythm. Such an approach allows a great deal of opportunity for exploration and tailored organization of a music collection by end users, which is the key goal of an automatic organization scheme.

One of the issues with genre discussed earlier was that some genres are ill defined, concerned for example with time or place rather than musical characteristics. It would be a simple matter to add meta-data filters for such properties to the classifier, enabling it to distinguish those genres as well. The important distinction is that this data would need to be provided by the user and cannot be extracted from the music itself.

Clearly, this sort of ideal classifier is a very long term goal, especially considering that melody and harmony features cannot be reliably extracted at the present time. Nevertheless, such a system would revolutionize the way people interact with their digital music collections. There may also be business applications, in that a robust system could dramatically improve the experience of shopping for music. The previous success with supervised clustering described above suggests that the current features for rhythm and timbre might be useful, as they are or with adaptations, in a context of clustering. In the next chapter, some popular timbre features are examined, in an effort to quantify their usefulness for classifying piano music in the "classical" genre. Chapter 4 will discuss some methods by which rhythm analysis of music files can be improved.

Chapter 3: Timbre Analysis

3.1 Purpose

One of the key elements of music is timbre, which can be defined as "...that attribute of auditory sensation in terms of which a listener can judge that two sounds, similarly presented and having the same loudness and pitch, are different" [38]. In musical terms, this refers to a number of properties of an individual sound: the structure of its attack and decay (also called envelope), the changes in dynamics over its duration, and its overtone structure. Brass music, for example, features overtones only of odd integer multiples of the fundamental frequency and the higher overtones take longer to emerge, resulting in a distinctive sound associated with brass instruments. Perceptually, the timbre of an instrument type is distinctive, even though there is variation in timbre between different versions of the same instrument. For example, different types of pianos have unique sounds; however, generally every type of piano can be readily identified as such by an untrained listener.

An effort to implement clustering of instrumental music directly based on timbre features used in the Tzanetakis research [14] and also used by other researchers failed. I chose to use my own implementation of the DBSCAN algorithm, which is described in [18]; data structures necessary to support an efficient implementation are discussed in [39]. In short, DBSCAN takes density and minimum cluster size parameters as input and finds clusters of arbitrary sizes, while classifying points that do not fit into the clusters as noise. It does this by looking for data points that can be grouped together into clusters containing more than the minimum number of points while not exceeding the overall density metric. In a test using a wide mix of piano and organ music, the program found two clusters, as expected, despite the presence of outliers. However, the results became worse as a greater variety of music types was added: the clusters tended to not be sufficiently far apart and the number of outliers tended to be too large, leading DBSCAN to find only one large group. This experience suggested the need for an investigation into the granularity of the feature set associated with timbre, with the goal of determining whether the features in question are independent of style, rhythm, and signal transformations that change the signal, without affecting human perception of the signal (transparent transformations).

3.2 Feature Overview

All of the following methods provide short-time features to describe an audio signal, meaning that to give good results they must be applied over very brief time fragments when the signal is very stable. As a result, a multilevel approach is taken in their extraction. The basic feature extraction operates at the lowest level, in the *analysis window*, which is 23 ms in length. The larger *texture window* represents a series of non-overlapping analysis windows, enough of them to define a musical texture as perceived by a human [14]. I have kept the Marsyas default value of 1s for the texture window, which corresponds to 43 analysis windows. The texture window is represented by means and variances of the features over all analysis windows contained in it. The analysis of a full audio file will result in means and variances of the means and variances of features in all the texture windows contained within a file. It is beneficial to analyze multiple texture windows simply because that strategy reduces the "bad-luck problem" of picking a texture window that is not characteristic of the whole file.

To summarize, applying this process to a whole piece of music provides four different statistics for each low-level audio feature (described beginning in Section 3.3), indicating both the overall information about average feature values throughout the file, as well as information on their variation from one texture window to the next. For ease of writing, each version of a feature will be distinguished by a prefix: aa-, as-, sa-, and ss-, where aa means "average of average," as means "average of standard deviation," and so on. The following two controls are applied to the audio files in the experiments:

- The first thirty seconds are skipped in order to avoid initial non-musical sounds, such as applause.
- The files are trimmed to fifteen seconds to both eliminate nonmusical sounds at the end of the file as well as control for differing lengths of different pieces of music. Longer segments did not change feature content significantly.

The above controls in the feature extractor result in a total of 15 texture windows with 645 analysis windows. The actual test data will be described following an overview of the features extracted by Marsyas. The basic flow of the analysis-window fragment of the signal through the series of timbre-extraction MarSystems is illustrated in the

following diagram. Each box represents a MarSystem (or, a collection of MarSystems), labeled and followed by a description of inputs and outputs. In short, the flow changes from a signal $s(t)$ to a frequency spectrum $S(f)$ and finally to feature values given by the subscripted Xs, which are collected into a list and used for classification. There are thirteen total MFCC features and four other features. For each feature type, four values are extracted, for a total of sixty-eight features in the complete feature vector. Subsections following the diagram provide the mathematical details of the processes depicted.

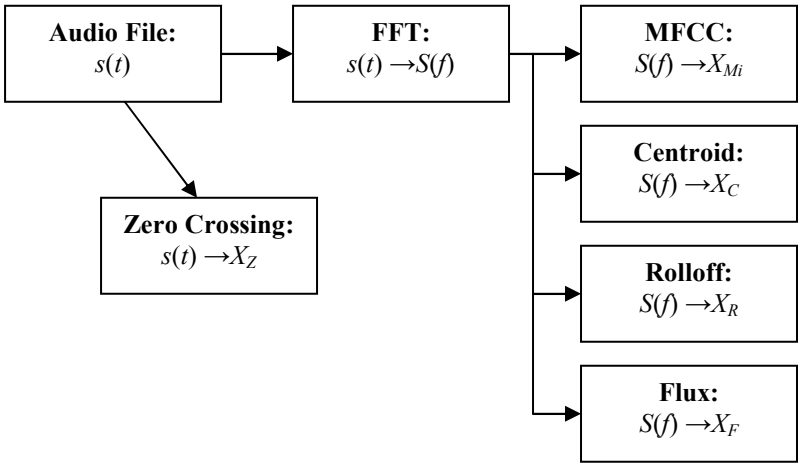

Figure 3.1: Feature extraction process

3.3 Mel Frequency Cepstral Coefficients

The Mel Frequency Cepstral Coefficients (MFCCs) are derived from the Fourier transform of the audio signal adapted to the Mel scale, which is a perceptually-determined pitch scale that corresponds to human hearing. Below about 1000 Hz, humans perceive a doubling of frequency as a doubling of pitch, but above this point the association becomes logarithmic. The Mel scale [6] approximates this relationship as shown in the following conversion between frequencies in Hz (f) and Mels (m):

$$m = 1127.01048 \cdot \ln\left(1 + \frac{f}{700}\right) \tag{3.1}$$

The MFCCs are determined by applying a Discrete Cosine Transform (DCT) to each of a series of Mel filter bank energies, resulting in a series of coefficients, of which the first thirteen are used [5] as Z_{M1} to Z_{M13}. Appendix A contains further details on the procedure. Tzanetakis and subsequent researchers typically use only the first five coefficients for music genre recognition [14]. It is reasonable to suppose that subtle differences in instruments might be detected with the use of more coefficients, so thirteen were extracted for these experiments.

3.4 Short-Time Spectral Features

The following features are based on the magnitude spectrum *S*(*f*) that results from the Fast Fourier Transform of each short analysis window. They are referred to as the spectral features:

Centroid

The Spectral Centroid is the center of gravity or second moment of the spectrum. It provides a measure of the shape of a spectrum; higher values have more high-frequency components and vice-versa. The equation for the centroid is:

$$X_C = \frac{\sum_{f=1}^{M} S(f) \cdot f}{\sum_{f=1}^{M} S(f)} \tag{3.2}$$

Rolloff

The spectral rolloff is the frequency below which 85% of the magnitude of the spectrum is contained. It also provides a measure of the density of low frequencies in the signal. Mathematically, the rolloff is X_R in the equation:

$$\sum_{f=1}^{X_R} S(f) = .85 \cdot \sum_{f=1}^{M} S(f) \tag{3.3}$$

Flux

The spectral flux measures the difference between spectra of subsequent time frames. It is defined as:

$$X_F = \sum_{f=1}^{M} \left(S(f) - S(f-1)\right)^2 \tag{3.4}$$

Time Domain Zero-Crossings:

For convenience, this feature is grouped with the spectral features, although it is actually extracted from the time-domain signal. Zero-crossing is simply a count of the number times the signal crosses from positive to negative or vice-versa.

The complete feature vector contains sixty-eight values, consisting of four statistics (averages of averages, averages of standard deviations, etc.) for each of the four spectral features and for each of the thirteen MFCC coefficients.

3.5 Experimental Method

The primary difficulty with clustering is the selection of features useful in distinguishing the kinds of groups desired, without prior information about what all the possible groups are. The starting point of the following experiments is a simplified

timbre classification problem, one which involves only non-vocal music using one instrument. With this restriction, the problem of timbre-based clustering as described here simplifies to a problem of instrument identification.

The recorded repertoire for piano dwarfs that of any other solo instrument, and represents an obvious starting point for investigating feature variation due to instrument details, performer and style. From the author's music library, seven groups of piano music were chosen, each of a particular style. The emphasis was on short, characteristic pieces that evoke a style, rather than highly unique masterpieces. For comparison, groups of violin and cello music were also collected. Each group has about fifteen files. A complete list of files and their associated label and catalog number may be found in Appendix B.

- Group 0: A collection of piano preludes by Cesár Cui. They are simple and similar in their Romantic-era style. All are played on the same piano by the same performer. This forms the "base group" with which comparisons will be made.

- Group 1: Brief piano preludes by Fryderyk Chopin. These are also fairly simple in construction and in a Romantic-era style. They are all performed by the same person on the same piano.

- Group 2: Short, simple Romantic-era piano works by composers such as Chopin, Amy Beach, Sergey Rachmaninov, and Anton Bruckner. They are stylistically and rhythmically different, but fall within the field of evocative romantic piano music. They are performed by multiple performers on multiple pianos.

- Group 3: Performances by pianist Glenn Gould of the Goldberg Variations by Johann Sebastian Bach. These are works originally for harpsichord and feature a style distinct from the Romantic music above, but they are melodic and pleasing to the ear. They feature Gould on the same piano throughout.

- Group 4: Avant-garde piano music in a variety of atonal, serialist and aleatoric styles by composers such as Arnold Schoenberg, John Cage, Karlheinz Stockhausen and Pierre Boulez. These files tended to be outliers in the DBSCAN clustering effort. Their style is unconventional, harsh and non-melodic, but nevertheless they are obviously piano music. This group features different performers on different pianos.

- Group 5: A collection of solo cello suites by Bach, featuring the same performer on the same instrument. This represents a best-case scenario for comparison with Group 0.

- Group 6: A collection of Romantic-era caprices for violin by Otto Paganini, featuring three performers on multiple instruments. The style is virtuosic but melodic across all the caprices.

As mentioned in the discussion of feature extraction, the first thirty seconds were ignored, and the subsequent fifteen seconds were analyzed. Every file was extracted from CD without apparent error, and compressed using the `--preset-standard` option of the LAME encoder. According to a series of double-blind listening tests, this setting produces transparent compression to the author's ears. Table 3.1 depicts the relationship between files in the above groups; italicized groups feature non-piano music.

Table 3.1: Audio file groupings for classification; non-piano groups are italicized

	Within-Group Performance Variety	
	Low	**High**
Difference in Style (from Group 0) Extreme		Group 4
Medium	Group 3, *Group 5*	
Low	Group 1	Group 2, *Group 6*

The distribution of values for each group were analyzed to find the mean and variance, and obvious outliers were identified and removed, including certain files which for unknown reasons were incompatible with the Linux-based MP3 decoder used by Marsyas. Because the values for each group appeared to be normally distributed, the Student's t-test was used to determine whether there were significant differences (at a 5% significance level) between the various tested groups. As a final test, the features extracted from compressed and uncompressed files were compared. Any features that changed significantly in value may be considered suspect. The lengthy raw data extracted from each group is supplied in Appendix C. Note that in the case of spectral features, Marsyas provides the values divided by an implementation-defined maximum possible value.

3.6 Same Style, Same Instrument Experiments

Comparing the base group, Group 0, to the two other Romantic-era piano groups provides an indication of which features are significant for defining Romantic-era piano

music. As the comparisons expand into broader categories and into other instruments, features which are insignificantly different should become significant. Group 0 and Group 1 represent similar styles, but with no within-group variety of performer. Using the notation introduced in Section 3.2, significant differences between these groups include only seven values: aaMFCC5, 9 and 11; saMFCC 10, 11 and 12; and saMFCC 10. These values can thus be considered prone to error due to variations in the sound of different pianos.

Features that are significant between Groups 0 and 2 suggest features that are especially unique to Group 0 as compared to typical values for piano music overall. Features that were significantly different between Groups 0 and 1 may become insignificant between Groups 0 and 2, due to the higher variance within Group 2. In fact, this is the case for all of the previously significant features. However, additional significant features appear in this test, indicating features that Group 0 and 2 share that are distinct from piano music generally. These features include aaMFCC 12 and ssMFCC 4, 6, 7, 10 and 12. It is particularly surprising that several of these features are of the ss type, indicating that the Group 0 pieces differ between texture windows more than Romantic piano music generally does.

3.7 Different Style, Same Instrument Experiments

The baroque style of Group 3 is distinguishable from the Romantic style of the pieces from Group 0, while the music from Group 4 takes the difference to an extreme. Because of this, these groups should have more significantly different features than resulted from either of the previous two comparisons. The 31 significant features comparing Group 0 to Group 3 bear this hypothesis out, as do the 34 significant features between Group 0 and Group 4. In both cases, over one third of these differences occur in the ss values. It's not immediately clear what this means, other than that the different styles of music vary quite a bit in their large-scale texture. This could be a consequence of the number of notes in each piece.

3.8 Different Style, Different Instrument Experiments

Features which become significant between Group 0 and Group 5 but which are insignificant in the other three are features characteristic of the piano sound, regardless of how it is played. There are 8 of these features, including aaMFCCs 1 and 8, asMFCCs 6 and 7, and saMFCCs 5 and 10. It's worth noting that none of the spectral features are significant in this test; considering their low-level nature, this does not seem surprising.

3.9 Same Style, Different Instrument Experiments

The preceding comparison between Group 0 and Group 5 represented a best-case scenario, in that the difference between the groups was designed to be as large as possible, whereas the variation within each group was designed to be small. The comparison between Group 0 and Group 6 represents a more difficult scenario, with similar Romantic styles but different instruments. The breadth of performances is also wider. A total of 25 features were significant in both this and the preceding test, 28 features that were previously insignificant became significant, and the opposite happened only seven times. This is a somewhat confusing result because based on the design of the experiment, a broader, similar-styled group would be expected to have fewer significant values, not more. An explanation may lie in the fact that Group 5 features a cello, while Group 6 features violin music. Because they are both bowed string instruments with different pitch ranges and are expected to have very similar timbres, this might suggest that the timbre features are too dependent on pitch overall.

In any case, the values that remained significant between this and the previous test, while being insignificant in the earlier comparisons between piano groups, can be considered robust. These features are aaMFCC1; asMFCC5, 6 and 7; saMFCC3 and 10; and ssMFCC1. Note that these features are only robust in the sense that their values seem associated with the fundamental sound of a piano, rather than the subtle variations between different pianos. They are also distinctly different from the violin and cello groups, though whether clustering will distinguish between the groups with high accuracy depends on the number of outliers in the groups.

3.10 Patterns Observed in Test Results

One goal of these experiments was to find features that were invariant among piano music but which changed significantly for non-piano music; these features are listed at the end of the previous section. A second goal is to look at those features which seem to change for reasons that correspond with perceptible changes in the groups of music being compared. The table below presents a tally of patterns of significance (`S`) and insignificance (`I`) over all the tests described in sections 3.6 to 3.9. Recall that in each test, all the feature values for two groups were compared, and significant differences (using the t-test) were noted. For example, the features mentioned earlier as robust would have the pattern `IIIISS` because they are insignificant for the first four tests (the comparisons between different types of piano music), but significant in the last two (the comparisons between piano, cello and violin music). The number in the right hand column indicates the number of features which exhibited that pattern in the sequence of tests. Not all features varied in a consistent way, so many patterns and thus many features are not accounted for in the table, but the high occurrence of certain patterns suggests that they are particularly important.

Table 3.2: Differences in mean between groups for each test (S = significant, I = insignificant)

Pattern #	Significance Pattern	Count
1	IIIIIS	7
2	IIIISS	7
3	IIISSS	7
4	IISSIS	7
5	IIISII	4
6	IIISIS	4
7	IISIIS	4
8	IISISS	4

The second pattern is simply the set of robust features, and the third pattern represents features that are usually good for differentiating piano from cello and violin, but which fail in the fringes of extreme piano music. Other patterns are more challenging

to explain. The first pattern in the list suggests that the sound of the cello music is closer to the piano music than to that of the violin, even though one would intuitively suppose the cello to sound closer to a violin as they both feature bowed strings. A possible explanation is in the fact that a piano operates at multiple pitches which might average out to a cello-like pitch level, whereas the violin is uniformly higher in pitch; this is a problem in itself since ideally timbre features should be pitch-independent. The fourth pattern is even more confusing, but perhaps it can be thought of simply as a variation on the first pattern, with features that break down for non-Romantic piano music. The fifth seems to simply describe features that are more concerned with melodic music and fail only in extreme cases, with the sixth being a variation of the fifth in the same way as the fourth is a variation of the first. The last two patterns seem to defy reasonable explanation, except to say that the significance in the test between Group 0 and Group 3 might be related to the strong rhythmic character of Baroque music. Alternatively, those patterns might simply result from features that are unpredictable in their response to different instruments.

It is disappointing to note that only one of the robust features was in the aaMFCC set, since this group is conceptually the least dependent on the rhythm and melodic character of the music. The other robust features were in the asMFCC set and the saMFCC set, which may result from the fact that cello and violin are capable of sustained tones, whereas a piano is not (at least not to the same degree). It is also worth noting the poor performance of spectral features, none of which were shown to be robust. This does not seem too surprising because they may be highly correlated to pitch content, which varies enormously between different works. The zero-crossing rate proved to be equally lacking in value, because it is directly related to the noise inherent in each recording. A less naive approach to its extraction might overcome that problem.

Figures 3.2 and 3.3 contain two graphs highlighting two of the best features. They indicate how audio files from each of the tested groups cluster together in the feature space. Furthermore, the graphs suggest why the DBSCAN clustering failed: although there is a significant difference in the means of the string groups compared to the Cui preludes, there are still two notable outliers (note the triangles from Group 0

mixed among Group 5's squares), and the number of outliers only increases as a wider variety of piano files are added to the graph.

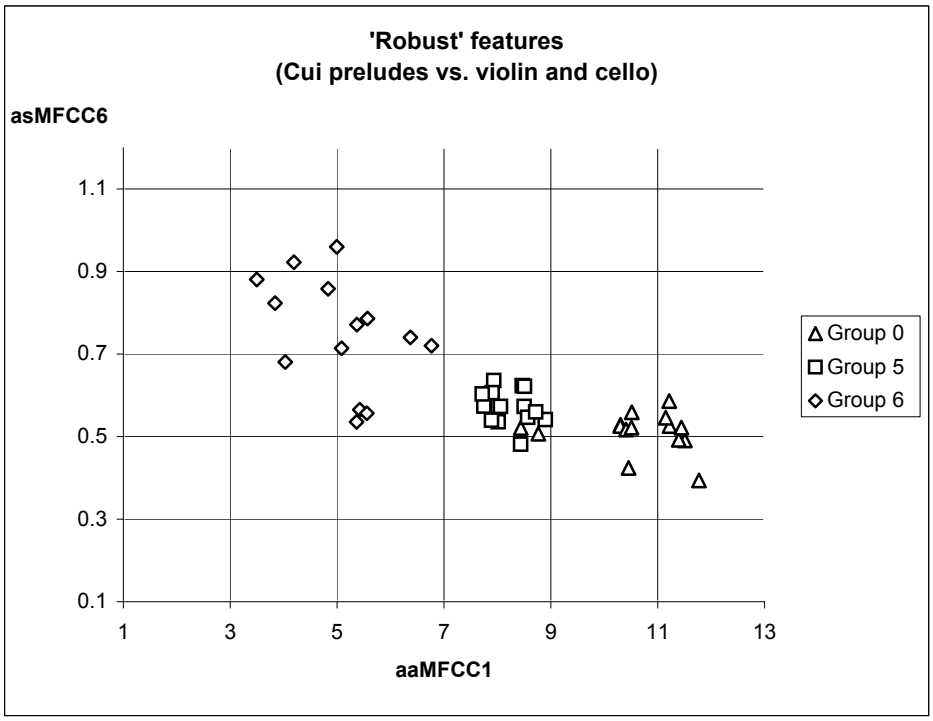

Figure 3.2: Best-feature graph for Cui piano preludes, plus violin and cello files

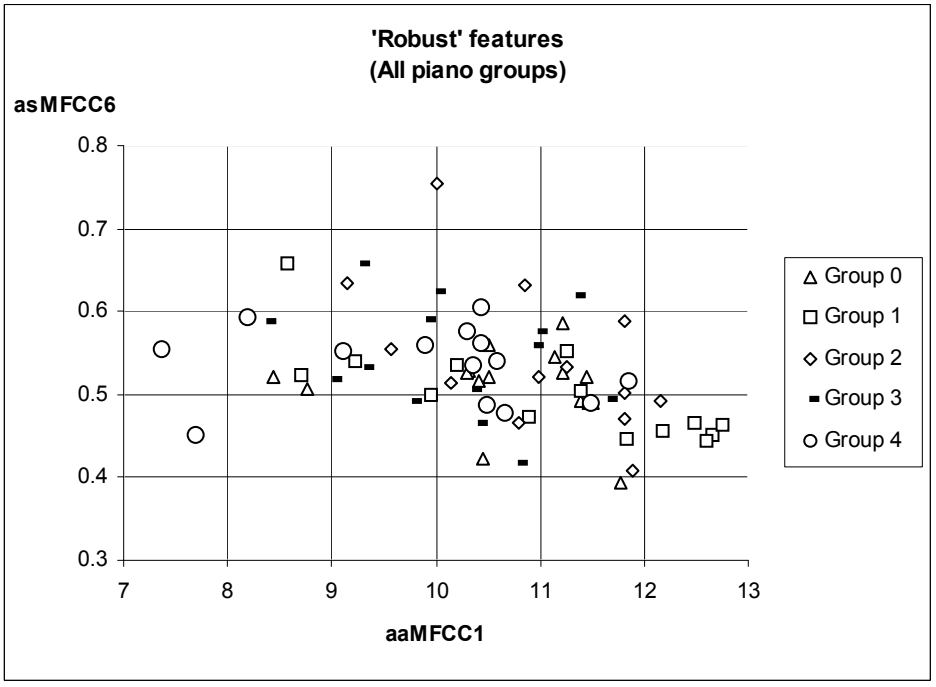

Figure 3.2: Best-feature graph of piano music, all styles

3.11 Limitations

The primary limitation of the experiments is that the features found to be useful are not robust on a grand scale, because features useful to distinguish piano from violin and cello may not necessarily be able to distinguish other groups of music from one another. Nonetheless, this set of features represents a direction for research with a much larger library. Although clustering is premised on the idea that all groups cannot be enumerated, a large enough data set with wide enough variety should find a pretty good set of features for clustering a wide array of music based on timbre alone. In other words, it is intuitive to suppose that features that work on the obvious instrument groupings may also work on the less obvious ones.

Another notable limitation is the use of a window of music to extract features. This is problematic because choosing an 'unlucky' window could result in an outlier, even though extracting a series of windows mitigates the problem somewhat. For a concrete example, one need merely look at the second prelude by Cui in Group 0, which proved to have substantially different feature values even though it sounds quite like all of the other preludes. A number of methods can be used to avoid this problem, including the automatic elimination of silence (though this presents its own challenges) or the scanning of an entire audio file, followed by segmentation based on the continuity of texture. That is, if the texture suddenly changes, music after that point would be considered separately from music before that point.

3.12 Effect of Compression on Features

MP3 compression uses perceptual models of human hearing in an attempt to eliminate data from recorded music that is not perceived [40]. Various bitrates of encoding achieve different quality levels of output for different file sizes; levels of compression over 192 kbps usually produce transparency for average listeners, although the issue is hotly debated. The Hydrogen Audio forums [41] provide serious discussions of audio compression quality, with the use of double-blind listening tests encouraged. For the following experiment, the same piece of music (a simple Bruckner piano piece) was encoded with the LAME encoder's constant bitrate option at four different levels: 320 kbps, 192 kbps, 128 kbps and 32 kbps. If the extracted audio features are based on

human perception, they would not be expected to shift as a result of MP3 compression at high bitrates when compared to the feature values of the original uncompressed file.

The MFCC values remained close to the uncompressed values after compression. The only values that saw a change greater than 10% at the 320 kbps level were saMFCC0 (10.8%), saMFCC11 (12.3%), ssMFCC1 (16.2%), and ssMFCC11 (24.3%). Values that changed more than 10% at the 192 kbps level included aaMFCC6 (11.3%), aaMFCC8 (10.5%), saMFCC0 (11%), saMFCC11 (16.4%), ssMFCC1 (16.8%), ssMFCC4 (14.4%), ssMFCC9 (15.1%), ssMFCC10 (12.3%), and ssMFCC11 (27.7%).

The spectrum values and time-domain zero crossing rates suffered significantly after compression; especially the standard deviation values, which change more than 1000%. The direct average values, aaCentroid, aaFlux, and aaZeroCrossings remain within 10% of the WAV values until compression goes all the way down to 32 kbps. Interestingly, the aaRolloff feature experienced less than 1% change at all compression levels, including the most extreme. Table 3.3 specifies how many MFCC and spectral features changed by different percentage levels at different bitrates:

Table 3.3: Number features (MFCC, spectral) at various bitrates that changed a specified %

MP3 kbps	< 1%	1-5%	5-10%	> 10%
320	8, 4	27, 3	13, 2	4, 7
192	9, 2	26,2	8, 4	9, 8
128	9, 3	25, 3	10,0	9, 10
32	1, 1	5, 0	3, 1	43, 14

Overall, these results suggest that MFCC features are generally consistent even after high levels of compression, whereas the standard deviation values in the spectral feature set should be treated with much more skepticism.

3.13 Conclusions

The above experiments highlight a handful of features that proved useful in distinguishing piano music from that of cello and violin, and also provided an exploration of the way in which features change as the stylistic and instrumental variety of a group of

music changes. The MFCC features proved overall to be the most useful for distinguishing significant differences in the means of groups of piano and string music. They also appear to change only slightly as a consequence of MP3 compression. New features will be needed before unsupervised classification can be very effective, however, because of the variation of many of the features as the range of piano music expanded, and the fact that even the best features seem to produce a large number outliers when piano music is considered as a whole.

Chapter 4: Rhythm Analysis

4.1 Purpose

Rhythm, one of the fundamental properties of music that can be used for classification, is the organization of musical sounds in time. In written music, the organization of beats is often specified by a time signature, which is a set of two numbers that state the relative size of a beat in terms of note value, and the number of beats per measure. This is useful information during performance and certain beat patterns are often associated with particular time signatures. However, it is possible to subdivide the time signature beats of a measure into multiple distinct sound units, and almost any pattern of sound units can be represented within a measure of any time signature. Consequently, the identification of time signatures through signal analysis is not a useful way to approach rhythm analysis.

Most of the previous work in rhythm analysis is related to tempo identification. Tempo refers to the speed with which repetitive sound units occur, and tempo extraction and matching is extremely useful for disc jockeys who are programming dance music mixes. A number of strategies have been employed in the past, including identification of low-frequency pulses [8]. Tzanetakis introduced the concept of a "beat histogram" [14], a tool which is created through the use of the signal autocorrelation. Its purpose is to find the repeating signal patterns in music that define a rhythm. Foote and Uchihashi use a similar strategy to analyze the rhythmic structure of music in [24], but their efforts are more geared towards visualization that the extraction of features for classification.

I have chosen to focus on classical music, a genre which Tzanetakis claims can be separated from other genres by its lack of strong rhythms, throughout this thesis. However, virtually all composers in the past five hundred years wrote dance music, and music from the Baroque and Classical eras can be readily identified by their rhythmic character. Below is the histogram for an early Mozart piece (Koechel catalog number 33b) performed on flute, taken from the first *Rarities and Surprises* disc in Phillips' *Complete Mozart Edition*. The histogram was generated using Tzanetakis' own algorithm. The vertical axis represents the degree of presence of different periodicities or beats in the signal represented by Beats Per Minute (BPM) on the horizontal axis. In this

and future histograms, the relationship between different peaks is more significant than the significance values with which they are associated.

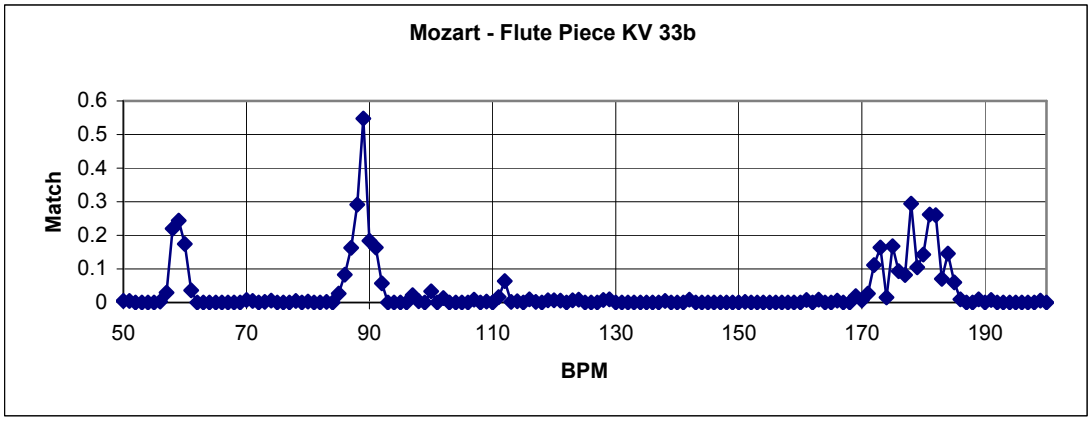

Figure 4.1: Beat Histogram for Mozart piece (KV 33b) showing different beats within the signal

The peaks in the histogram suggest a rhythm much stronger for this piece of classical music than the various examples Tzanetakis presented. For a histogram more similar to Tzanetakis' examples, see section 4.7. The true tempo from listening to the music is at the high end, towards 180 BPM, though it varies slightly and pauses at several points; the additional peaks at approximately 180/2 and 180/3 are the result of an echoing effect that will be discussed at the end of this chapter. This example indicates that rhythm remains a useful tool for classification in the real of classical music, even if it is not as obvious a surface feature as it is in hip-hop, rock music and other popular music.

4.2 Generating Beat Histograms

As mentioned above, beat histogram generation is done through using an autocorrelation strategy implemented in Marsyas, the feature-extraction tool introduced in Chapter 2. However, before the autocorrelation can be computed, nine previous MarSystems manipulate the waveform to prepare it. The first step is a shifting and downsampling step, which provides the input waveform in a gradually growing manner: the output of this initial MarSystem consists of the last output seen, along with a small new chunk of the waveform. Using the provided default values results in a waveform that is 6 s long, with

the last 37 ms consisting of audio data newly extracted from the file. Thus, the audio is read from the file with what amounts to a sliding window. This makes sense, as the newly added information is what will be used to compute the autocorrelation through comparison with the previous information. A graphical representation of the entire histogram-generation process is shown in figure 4.2. The signal $s(t)$ is decomposed by the Discrete Wavelet Transform (DWT) and recomposed into four signals using the Inverse Wavelet Transform (IWT) representing different frequency bands of the original. Each of these undergoes an envelope extraction stage before they are summed together into the envelope signal, $s_e(t)$. The autocorrelation of s_e is then computed, which indicates how well the signal matches itself at lag time l. Next, a peak-picking process chooses three peaks, associates these tempos with their associated Beats Per Minute (BPM) values, and adds them to the histogram h(BPM). Further detail on each step follows the diagram. The whole process operates on one window at a time, though each succeeding window includes information from the last window in order to identify tempos that cross window boundaries.

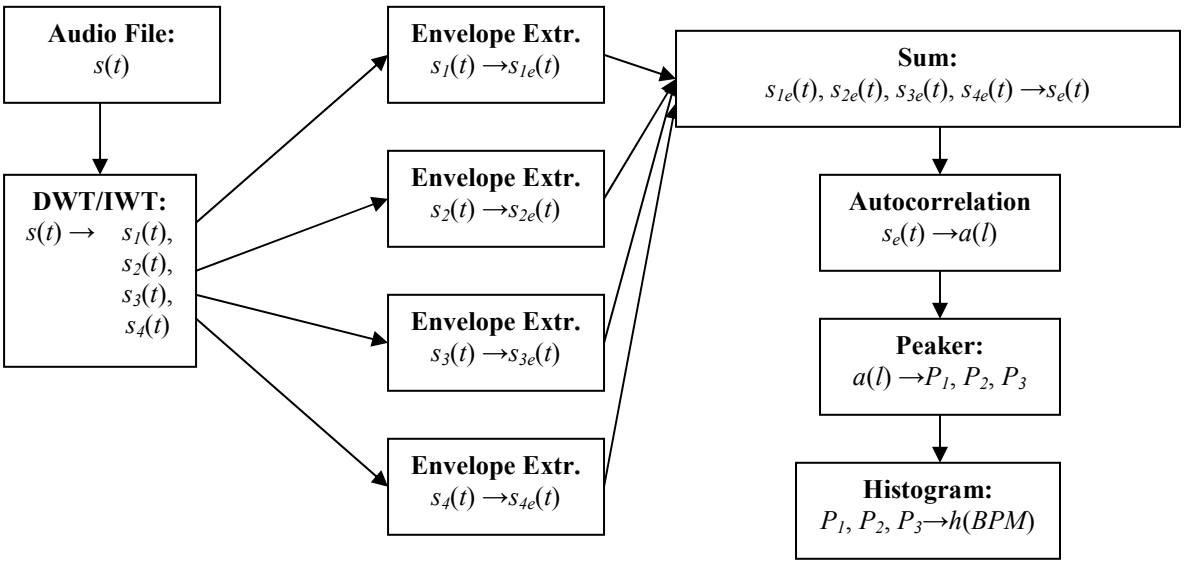

Figure 4.2: Data flow diagram for beat histograms

Discrete Wavelet Transform

The initial step is a Discrete Wavelet Transform (DWT), which is a tool for signal analysis similar to the Fourier transform, but which provides both time and frequency information. The DWT is generally computed using a pyramidal algorithm, and is often

used for image compression. In general, the idea is to decompose a signal into multiple components of different frequency and temporal resolutions: high temporal resolution but low frequency resolution at high frequencies, and low temporal resolution but high frequency resolution at low frequencies. The decomposition is achieved by successive high and low pass filtering. It is then possible to rebuild the signal, in its original or a filtered form, at a later time by integrating some or all of the decompositions using the Inverse Wavelet Transform (IWT). By integrating particular bands, arbitrary resolutions or quality levels for the result can be achieved, hence the popularity of the DWT in compression. More details on the DWT can be found in [45].

In the Marsyas implementation, the pyramid-based DWT is used to break the input signal into an octave decomposition. The IWT recomposes the decomposition into four signals in the time domain, each containing only the information from the octave-sized sub-band from which it was built.

Envelope Extraction

Each of the four signals produced from the previous DWT/IWT step undergoes an envelope extraction process, followed by normalization and downsampling. First, the signal is passed through a full wave rectifier to invert the negative portions of the signal, and then it passes through a low-pass filter using a one-pole implementation. These steps are frequently used for the purpose of smoothing the signal envelope. Based on the variables used in the diagram above, this process is described mathematically by

$$s_{ne}(t) = .01 \cdot |s_n(t)| + .99 \cdot s_{ne}(t-1) \tag{4.1}$$

This extracts the amplitude envelope of the signal. Next, the result is downsampled and normalized to center it about the zero axis. The amount of downsampling determines the granularity of the peak-picker described below. After downsampling, all four of the signals are summed back into one.

Autocorrelation and Peaking

The autocorrelation of a signal can be used to detect periodicities. The mathematical definition of the continuous autocorrelation of signal $s(t)$ is:

$$a(l) = \int_{-\infty}^{\infty} s(t)s^*(t-l)dt \tag{4.2}$$

The $s^*(t)$ represents the complex conjugate of $s(t)$. In practice, he autocorrelation is computed efficiently by using the Fast Fourier Transform (FFT), based on the algorithm provided in [42]. In short, the FFT of the signal is multiplied by its own complex conjugate, and the result undergoes an inversion. The output is a vector that represents the degree to which the waveform matches itself for a given lag time. If plotted on a graph of correlation amount vs. lag time, the output would look like a series of hills, and thus it is necessary to use a peak-detection function to find the clearest periodicities. In the Marsyas approach, a window is defined, and only one peak value is chosen within each window. This avoids picking more than one value per hill. Internally, the periodicities are represented as numbers of samples. In order to convert the lag value l given a sample rate r into a Beats Per Minute (BPM) value, the following equation is used:

$$BPM = \frac{60r}{l} \tag{4.3}$$

Note that as sampling rate goes down, the granularity at higher BPM is reduced. This is because each step in lag value results in a progressively large step in BPM; once the BPM step size is greater than one, holes in the graph will be observed. As a result, a smooth graph of BPM versus intensity of beat requires that the signal not be substantially downsampled before the autocorrelation is performed; the downside is that the autocorrelation takes longer to compute.

Histogram

The autocorrelation is only performed over approximately six seconds of audio, which is reasonable since extracting extremely low-BPM periodicities is a bit silly. I

follow the Tzanetakis approach by looking at tempos between 50 and 200 BPM; six seconds serves this range well because several instances of a 50 BPM tempo can occur within one window. Because most files are longer than six seconds in length and because most real music does not have a consistent rhythm at all times, the entire file is analyzed. To find the most common rhythmic component, a histogram is built by converting the peak values to BPM and adding them to a histogram which is accumulated over the entire file. The resulting histogram gives the relative intensity of each possible tempo in the music. The Mozart example above was generated by using precisely this process.

4.3 Limitations of General Beat Histograms

The beat histogram is typically extracted over the course of an entire file; if the music is sufficiently rhythmic, the steady sections of rhythms will overpower the sections of variations (in popular music, typically the intros, outros and bridges have a different rhythm from the main song), thus producing a histogram with a few sharp peaks. A big problem occurs in pieces of music in which the rhythm undergoes a substantial change during the piece: the net result will be a histogram lacking salient peaks. There will be peaks implying the presence of multiple rhythms when, in fact, some rhythms were never observed together, or there will be so many occurrences of so many rhythms that no clear peaks will be noticeable. Even in the simple case of music containing only two uniform tempos, it would not be clear from the full histogram if these tempos occurred simultaneously, or if they occurred in two distinct halves of the music. Due to its increased complexity, this problem is especially common in classical music, since many classical works have more than one movement or have multiple tempos per movement.

To help solve this problem, I have created a system of histogram tracking, which is similar to the beat trackers like the one described in [8], but in this case an entire histogram is tracked, and all rhythmic components can be seen at all times. The tracking component uses previous observations to build a gradually-changing "running histogram." I then discuss a novel method of retrieving features from such histograms.

4.4 The Histogram Tracker

Normally, histograms are accumulated over the entire audio file. In a digital audio signal, a single frame is the signal value that occurs at some point in time, and the frame rate defines the maximum frequency which can be represented. The granularity of the resulting histogram depends on the sample rate *r* of the input signal, while the slowest detectable repetition depends on the size in samples *n* of the audio section over which the autocorrelation is computed. As a consequence, the histogram goes through subsequent accumulation stages every *n* samples, slowly building up a histogram across the whole file. Each of these stages is a single iteration of the same process, resulting in a partial histogram for that particular set of *n* samples. In contrast to the usual approach, I use these partial histograms to build up a "running histogram" every αn samples, where α is an adjustable integer that indicates the number of iterations to go through before updating. In other words, a new histogram will be generated every $\frac{\alpha \cdot n}{r}$ seconds.

The histogram tracker is useful for finding transition points where the rhythm changes suddenly, but sometimes a rhythm might vary slowly during a performance. It would be best if the histogram tracker could avoid false positives by incorporating data from previous histograms, so that miscellaneous errors in the beat (whether caused by bad recording or by performer choice) are smoothed over. To this end, the running histogram represents a weighted summation of the four most recent partial histograms for each of the four most recent iterations. The weights were chosen so that the influence of the most recent sample is high, and the influence level decays rapidly down to zero. The value stored in the running histogram at BPM value *i* in the running histogram is represented as follows, where *t* refers to the present iteration:

$$h_{running}[i] = 1 \cdot h_t[i] + .5 \cdot h_{t-1}[i] + .2 \cdot h_{t-2}[i] + .1 \cdot h_{t-3}[i] \tag{4.4}$$

This strategy produces a histogram that slowly changes as the beat of the music changes. The frequency of false positives of detected incorrect tempos decreases as the value of α increases, but it will take longer for a true change to be observable, and vice versa. This is the basic edge-detection tradeoff. The parameter α should be, at a minimum, the time it

takes before a particular tempo would be considered stable. In practice, I use a value of 12 so that a new histogram is generated approximately every four seconds, given the default Marsyas setup which downsamples files to 22.05 kHz and analyzes using n = 1838. Four seconds also fits well with the minimum BPM detectable by the default Marsyas setup of 50 BPM. Four seconds is long enough to allow a bit more than four beats of a 50 BPM rhythm, and more of higher-BPM rhythms. These values are used in all the experiments described in the following sections.

4.5 Histogram Features

A quick look at the beat histogram shown for the Mozart piece reveals that it is reminiscent of a 1-dimensional mixture of normal distributions. I use this observation as a starting point, and apply an expectation-maximization scheme within Marsyas to estimate the parameters of a set of three Gaussian curves that best fit the current running histogram. The expectation-maximization algorithm will be reviewed as it applies to these one-dimensional histograms, and then I will describe the implementation choices made to account for the peculiarities of the application.

The Expectation-Maximization (EM) algorithm is an iterative-improvement algorithm used in pattern recognition to fit known distributions to observed data. Further details and applications can be found in [13] or in many statistics textbooks. The strategy can be applied to any distribution, depending on the user's intuition as to the form of the data. The histogram data extracted with the histogram tracker can be modeled as a mixture of normal distributions, where each distribution is normal with mean μ and variance σ^2, as per the following probability equation:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \tag{4.5}$$

The EM algorithm proceeds in two phases: an expectation phase and a maximization phase. The expectation phase computes the likelihood that each histogram point belongs to each of the distributions in the mixture model. The estimation step then re-computes the expected parameters (means and variances) of each distribution based on the likelihood values for each histogram point. In other words, each histogram point belongs

a certain percentage to each of the distributions, and the expectation step re-computes this value based upon the current tentative estimates of the distribution parameters

The likelihood l that BPM value i in histogram h belongs to Gaussian distribution j with mean μ, variance σ^2 and with the percentage of total values accounted for by the particular distribution as f is expressed as follows:

$$l_j(h[i]) = \frac{f_j\left(\sigma_j \cdot \sqrt{2\pi}\right)^{-1} \cdot e^{-(x-\mu_j)^2/(2\sigma_j{}^2)}}{\sum_k f_k\left(\sigma_k \cdot \sqrt{2\pi}\right)^{-1} \cdot e^{-(x-\mu_k)^2/(2\sigma_k{}^2)}} \tag{4.6}$$

The parameters of each distribution can then be recomputed. First, the percentage of the total likelihood accounted for by a given Gaussian is found:

$$f_j = \frac{\sum_i h[i] \cdot l_j(h[i])}{\sum_i h[i]} \tag{4.7}$$

The mean and variance are computed as one might expect, only modified to take account of the likelihood values:

$$\mu_j = \frac{\sum_i i \cdot l_j(h[i]) \cdot h[i]}{\sum_i i \cdot l(h[i]_j)} \tag{4.8}$$

$$\sigma_j = \frac{\sum_i i \cdot l_j(h[i]) \cdot (\mu_j - h[i])^2}{\sum_i i \cdot l(h[i]_j)} \tag{4.9}$$

The iterations repeat until the parameters converge, which is implemented as a maximum change over all distributions of less than 0.01 for means and 0.0001 for variances.

During testing, it was discovered that the strong, constant rhythms of synthetically generated beats produce activations at an isolated BPM value in the histogram, which causes the deviation to shrink to zero and ultimately generate divide-by-zero problems. In such situations the deviation is set to a very small non-zero value. This is primarily an artifact of the use of synthetic rhythm and the discrete nature of the histogram, not a flaw of the model choice.

Any further classification of audio signals based on these histograms, particularly on the means and variances, requires that there be a finite and constant number of distributions in the model. To that end, I have chosen to model only three distributions. This choice was made based on the observation that histograms mostly contain peaks that are multiples of one another, and only rarely are more than three simultaneous peaks observed in a running histogram, though in the original implementation, however, many more could be potentially observed in a file with many different rhythms. When there are only one or two peaks, two or more of the estimated means will be identical. The other distribution parameters also carry meaning. The variance indicates how stable the beat is, and the percentage *f* indicates the overall strength of that rhythm compared to others.

Initialization is a common problem for EM implementations, but due to the single dimension and the fact that peaks are typically far apart, initializing each distribution with means at the minimum, maximum and center of the histogram range has proven effective in practice.

4.6 Applications to Synthesized Rhythms

The histogram tracking and EM algorithms were tested using rhythms created by the Hammerhead Rhythm Station, a freeware beat generator [43]. In order to test the tracker's ability to recognize histogram shifts and produce correct EM values, 15 rhythm files in MP3 format were created and then fed into the system. One such file features one minute of an 80 BPM bass drum rhythm, followed immediately by one minute of a 180 BPM bass drum rhythm. The running histogram for each stable section, generated with the parameters described in the previous section, is shown in figure 4.3. Of particular interest are the multiple peaks in the second histogram, evidence of rhythmic echoing and an issue that will be discussed later. Also shown in Table 4.1 are the means discovered

for all three of the Gaussians with the EM algorithm. Note that the beat generator is not absolutely perfect, though the differences from the ideal are not noticeable to the ear.

The program also informs the user whenever it finds that one of the means has changed by more than 5 BPM. As can be seen, the detection of the two previously nonexistent rhythms occurred within one iteration, whereas the shift from 80 BPM to 90 BPM took about 10 seconds to fully stabilize; a lower value for the histogram length parameter α would certainly improve this performance. However, as long as the period during which the rhythm is stable is much higher than the time to stabilize, the problem doesn't matter very much with regards to segmenting the file for classification. For that purpose, whether a rhythm occurs in 45% or 50% of a file is not as important as the fact that it occurs at all. In this case, detection accuracy should take precedence over stabilization speed.

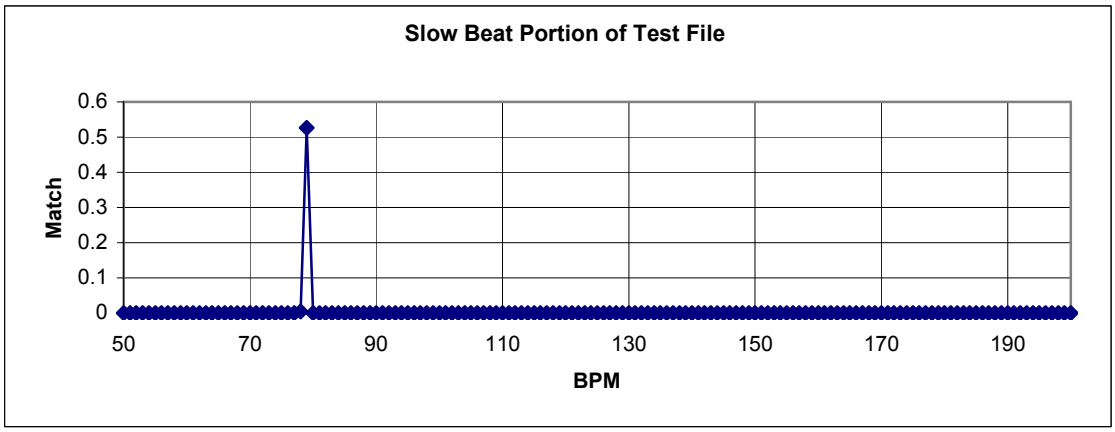

Figure 4.3: Histogram for slow part of test file (80 BPM)

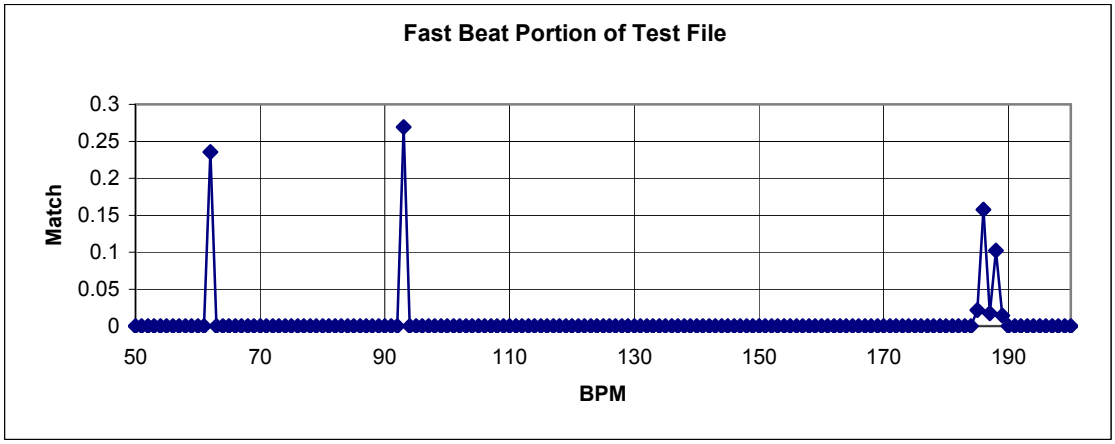

Figure 4.4: Histogram for fast part of test file

Time	Mean 1	Mean 2	Mean 3
4s	78	80	79
71s	62	n/a	187
75s	n/a	87	n/a
84s	n/a	93	n/a

Table 4.1: Changes in EM-generated means during run of file

Another test file featuring a steady rhythm with occasional skipped beats was also created to test the smoothing ability. There were no detected rhythmic transitions, so the smoothing appears to be successful and missed beats can be avoided effectively.

4.7 Applications to Real Music

The preceding sections discussed the method by which the running histogram operates, and its application to synthesized beats to prove its validity. Now it will be applied to actual classical music recordings to show how it compares to the traditional histogram approach. Consider Mozart's Flute Piece KV33b, which was introduced earlier as an example and which has a perceived tempo that is approximately 180 BPM. The original histogram, based on the entire file, and a sample running histogram are presented below. The running histogram is based on a section preceding the 44s point in the audio (total length is 74s). Since the music has no significant transitions between rhythms, the original entire-file based histogram works very nicely. The running histogram's only improvement is that the peaks in the lower ranges are a bit tighter, and the noise (such as the small peak at 111 BPM) has generally vanished. I should note again that this is a running histogram that would change throughout the file, with peaks shifting a little to the left and to the right as the performance varies slightly; the running histogram is representative of a stable section of the file, rather than a complete documentation of all changes. The values from EM estimation stabilized after about 12 seconds, and remain stable until the 53s point when a pause in the performance causes the center distribution's mean to be pushed just barely over the edge of a noticeable

difference. This suggests that the running histogram works, but the strategy for informing the user of transition points can still be fooled.

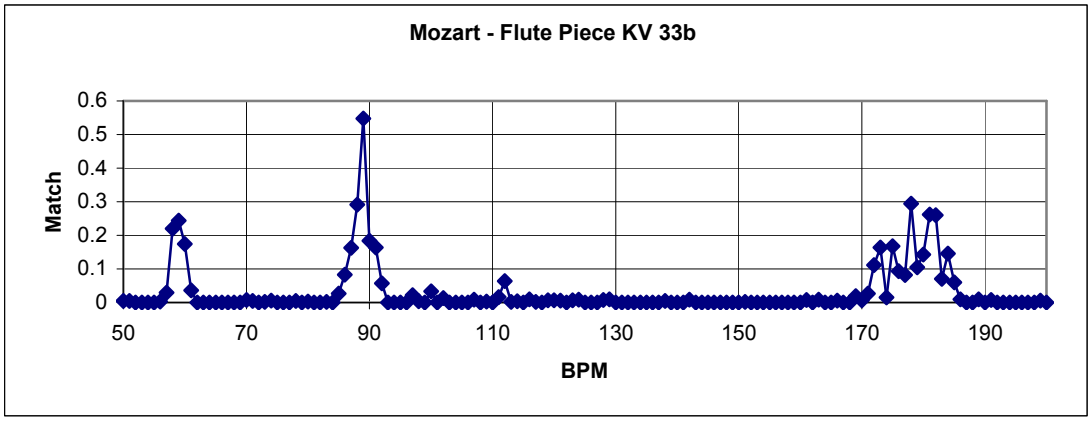

Figure 4.5: Mozart histogram, using original algorithm

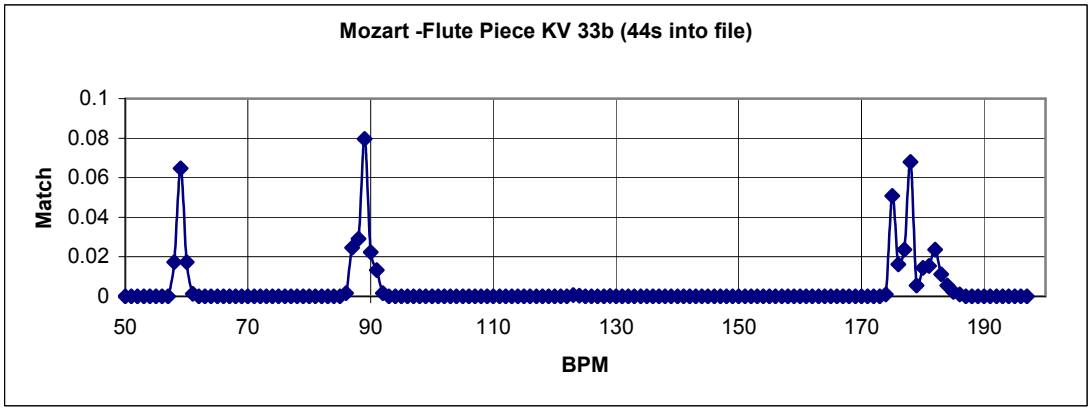

Figure 4.6: Mozart histogram, using data from 44s into the file

Next, consider a recording of an eight-minute orchestral waltz (the famous Kaiser-Walzer or Emperor Waltz) by Johann Strauss II. The music is characterized by an array of different rhythms and tempi, with extensive ornamentation and variation in between sections. It is a waltz in name only, and varies widely in its rhythmic structure. In short, this music represents an extremely hard file to characterize. As expected, the overall histogram for the whole file is extremely poor, showing no particular dominant rhythms.

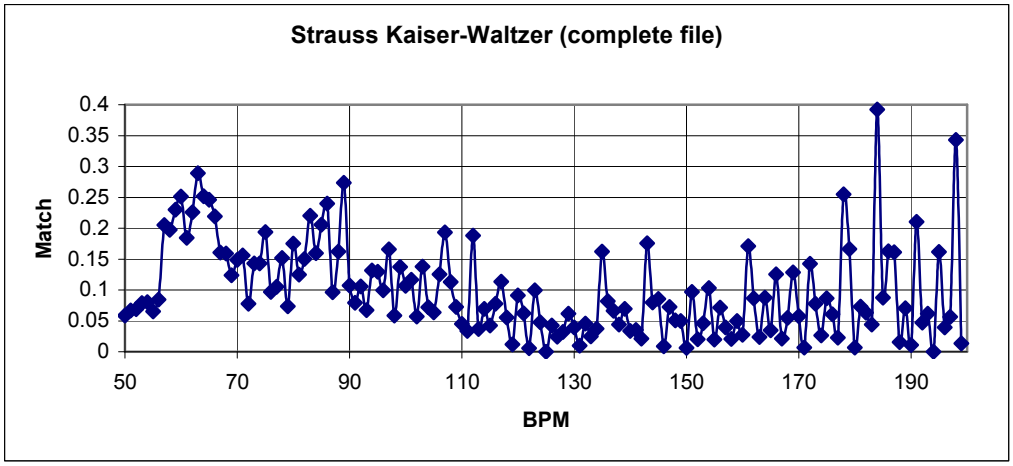

Figure 4.7: Strauss histogram, over the complete waltz

Three running histograms from different points in the file are show in Figures 4.8, 4.9 and 4.10 below. The first comes from a little before the 4:00 mark, at which point the music takes on a fairly slow character. One primary peak dominates at 59 BPM, which is approximately the observed tempo. The second two histograms represent the violin solo that begins around 1:40 and the more rhythmic section immediately following it. The violin solo is primarily slow, swooping tones without a strong rhythm, as evidenced in Figure 4.9, which has only one very obvious peak at 197 BPM, possibly corresponding to the tremolo (rapid, slight oscillation) of the violin. Also shown is a segment about 40 seconds later, when a rhythm has built up again; note the peak and surrounding "bulge" near 95-100 BPM, which again approximates the real tempo.

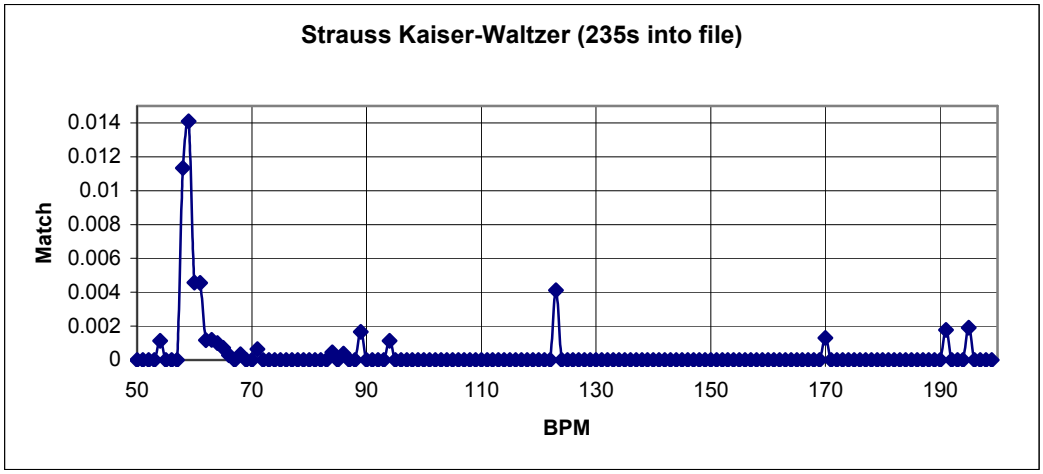

Figure 4.8: Strauss running histogram from 235s point

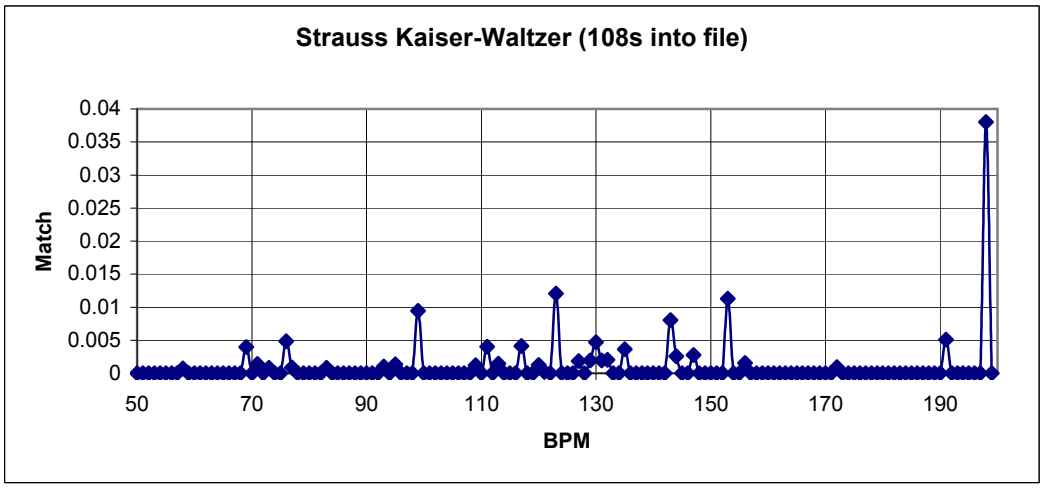

Figure 4.9: Strauss running histogram from 108s point

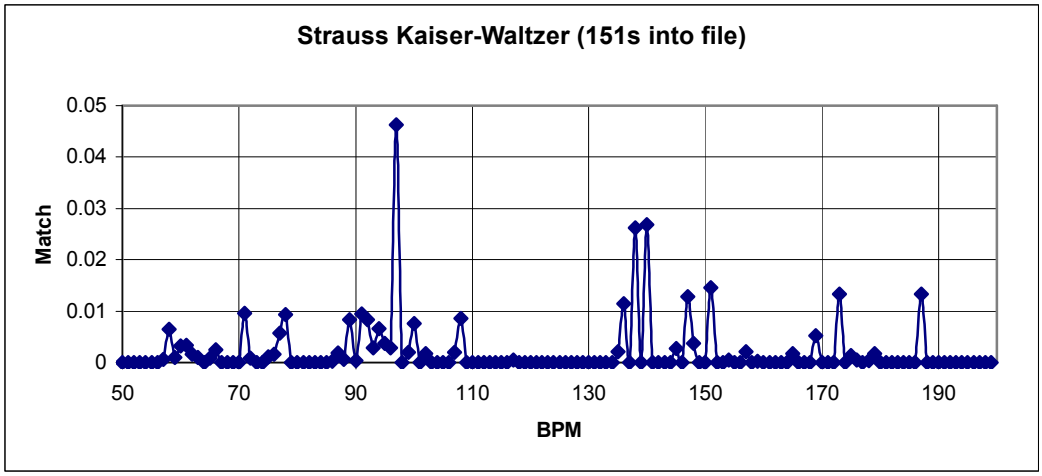

Figure 4.10: Strauss running histogram from 251s point

From the Strauss Waltz example, it is clear that a running histogram supplies some rhythm information in situations where the overall beat histogram is essentially worthless. However, the Strauss Waltz file exhibited shifts in tempo so rapidly that it cannot be segmented cleanly; this itself is a feature worth noting in a classifier. With a traditional beat histogram approach, the case of different rhythms being used for extended periods and the case of different rhythms used for very brief amounts of time can be indistinguishable.

4.8 Classifier Applications

The beat histogram tracker provides more information than a normal beat tracker because histograms contain information about multiple levels of rhythmic organization. The highest level is the sense of speed associated with a piece of music. This so-called "perceptual tempo" is different from the score tempo, which describes the speed of beats, not the speed of actual sounds, though they could be equivalent if the notes described by the tempo reference are those used predominately in the music. Additionally, care must be taken in human measurements of tempo, because the foot-tapping rate of a typical listener tends to be centered on the human heart rate [32]. One strategy to get this speed from a histogram is to choose the distribution with the highest mean that also accounts for more than 33% of the likelihood overall. For music which has multiple rhythms but can be segmented using the procedure described above, simply choose one value from

each segment. This will result in the file being classified in more than one group, which is perfectly reasonable. For rhythmically complex music, like the Strauss Kaiser-Waltzer example, no single value can be ideal, and segmentation is not practical. One choice is just the most commonly-observed mean (which is 167 BPM for that example, based on histograms from the entire file).

While the sense of speed is not too hard to extract from the histograms, the relationships between the means can be used to provide some clues as to rhythm. For example, a march (a repeating pattern of two beats, one emphasized and one not) would be expected to contain two tempos, one at k BPM and the other at $k/2$ BPM, whereas a waltz (constructed using three beats) would have tempos at k BPM and $k/3$ BPM. Such relationships are directly useful for classifying double-time versus triple-time music, and they can be found simply by comparing the EM-generated means. The problem is the echoing effect, because every rhythm that repeats at k BPM also will repeat at $k/2$ BPM, $k/3$ BPM, etc. If the waltz is constructed with two beats, one at $k/3$ and one at k, the faster beat will have an echo at $k/2$, for a total of three peaks in the histogram at k, $k/2$ and $k/3$. A march built with one beat repeating at k and another repeating at $k/2$ would also have an echo, this time at $k/3$ as a result of the faster beat. Thus, both histograms can look the same, depending on how the rhythms are constructed. Further work to force the autocorrelation to ignore such echoing would be very useful.

Having just described "different beats, same histogram" problem, it is important to note the "same beats, different histogram" problem. This one is more challenging than the other, because it relates to the way humans perceive sound, as opposed to the mechanics of the autocorrelation estimation. A beat that is perceived as a march by a person can be constructed by repeating one sound at k BPM and also repeating another sound at $2k$ BPM (just like in the above paragraph), so that the sounds overlap at every other beat of the faster sound. This might be represented as A—A+B—A—A+B... However, a rhythm can also be created that sounds very much the same to a human with two sounds repeating at k BPM, but shifted in time such that a single sound occurs at every $2k$ BPM, with the particular type of sound alternating. We could represent this pattern as A—B—A—B... To a human, this is still a march, but the beat histogram will get confused as shown in Figures 4.11 and 4.12. The basic problem is that it's looking

for signal similarities rather than noticing the "pulsing" at the 2*k* rate, so for rhythms constructed using alternating sounds, other methods of tempo identification will work better. Future efforts could revolve around combining different beat tracking strategies to solve this problem.

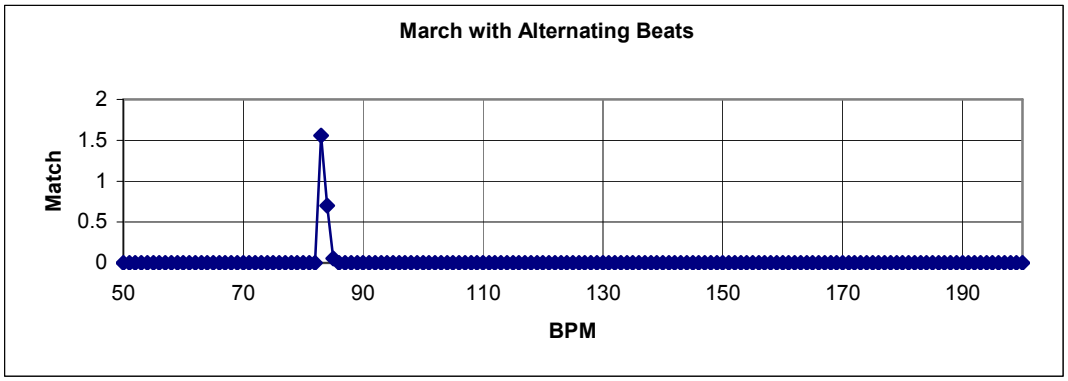

Figure 4.11: Histogram from march with two equal, time-shifted beats

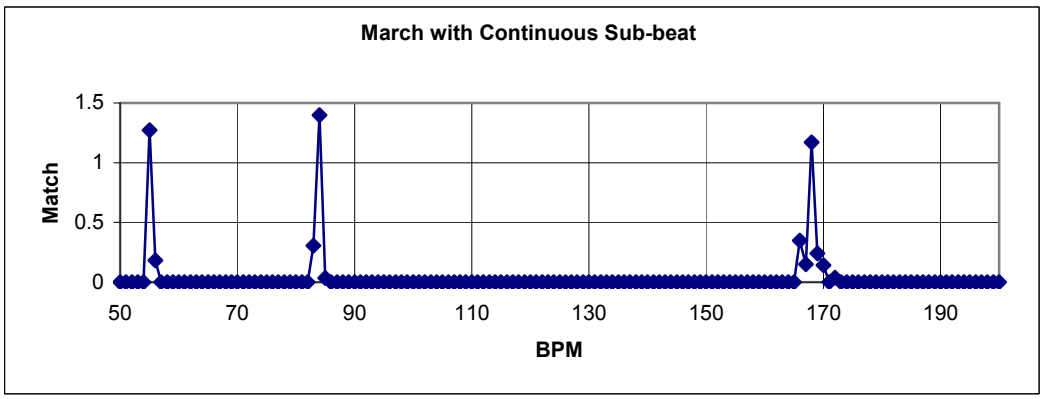

Figure 4.12: Histogram from march with two different, overlapping beats

4.9 Conclusions

The means of the various EM-detected distributions represent the tempos present in the music, and so they can be used directly as beat tracking. For audio classification, though, the relationship of the means is also of interest because these relationships represent rhythmic structure. Using such information allows classification based on both perceived tempo and rhythmic structure; however, detection of such relationships is subject to the caveats mentioned above involving echo and human beat perception.

Finally, the variance statistics can also be useful, as they represent the consistency of the beat in the music. Using the histogram tracker for segmentation can also allow for better classification of audio files by grouping files based on the actual steady rhythms present in them, so that a single file might be grouped into multiple categories. In short, the histogram tracking system and EM estimator fix some of the problems of the traditional beat histogram system, while highlighting some of the more challenging issues.

Chapter 5: Final Remarks and Future Directions

All of the preceding efforts in timbre and rhythm feature analysis serve the ultimate goal of unsupervised classification by showing, in the case of timbre, that new features are badly needed, and in the case of rhythm, by providing an algorithm by which features can be extracted from audio in a mathematically well-defined way. The fundamental challenge in classification is one of scale: music classification requires a broad level understanding of musical concepts, but at the signal level the features that are most commonly extracted are only indirectly related to the high-level properties. In the end, the problem will likely be solved by some kind of automatic music transcription system, but in the intervening years before such a system is created, there will need to be new, mid-level features extracted if classification beyond the extremely broad categorizations is to occur. Psychoacoustics will certainly have a role to play in the development of these new features.

In the domain of timbre, I believe that MFCCs have shown some promise, but may still be too dependent on pitch. Alternative strategies will need to focus on identifying the particular beginning and end of sounds within real-world recorded music, after which the instrument classification research that relies on isolated instrument sounds may become applicable to the general music classification problem. Additionally, it could be possible to normalize the pitch throughout an audio file such that only observations of timbre remain. The large scale timbre structure of music (concertos versus symphonies, for example) must also be tackled at some point. Perhaps a timbre histogram tracker of some kind could be applied to that problem.

In the domain of rhythm, the large-scale structure issue can be effectively tackled by the segmentation algorithm I present, with the caveat that it will work best in situations where the autocorrelation has already proven adequate in obtaining information about rhythm. Additionally, extraction of features using the expectation-maximization algorithm is better defined and more mathematically compelling than the ad-hoc peak finding strategies used in previous research. Future efforts in rhythm analysis should focus on strategies to integrate the autocorrelation approach with other beat tracking systems. It would also be quite valuable to identify characteristics shared by files for which the autocorrelation does not provide useful rhythmic information.

I believe that unsupervised music exploration and classification systems are in reach, and the use of such systems will add to the ongoing revolution in the way people interact with their music.

References

[1] C. J. C. Burges, D. Plastina, J. C. Platt, E. Reinshaw, and H. S. Malvar, "Using audio fingerprinting for duplicate detection and thumbnail generation," *Intl. Conf. Acoustics, Speech and Signal Processing*, vol. 3, March 2005, pp. 9-12.

[2] J. M. Batke, G. Eisenberg, P. Weishaupt and T. Sikora, "A query by humming system using MPEG-7 descriptors," *Proc. Audio Engineering Society 116th Convention*, Berlin, Germany, http://outpost.nue.tu-berlin.de/elvera/files/0787Batke2004.pdf, 2005.

[3] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content: a survey," *Signal Processing Magazine*, vol. 23, no. 2, March 2006, pp. 133-141.

[4] R. Clemmons, *Sounds in Time: A Guide to Music Fundamentals*, Kendall/Hunt Publishing Company, 1998.

[5] C. McKay, *Automatic Genre Classification of MIDI Recordings*, Master's thesis, McGill University, Canada, 2004.

[6] P. Cook, ed., *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*, MIT Press, Cambridge, 1999.

[7] R. M. Warren, *Auditory Perception: A New Synthesis*, Cambridge University Press, New York, 1999.

[8] E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-based classification, search and retrieval of audio," *IEEE Trans. Multimedia*, vol. 3, no. 3, 1996, pp. 26-27.

[10] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 29 no.2, 1981, pp. 154-272.

[12] S. A. Abdallah and M. D. Plumbley, “Unsupervised analysis of polyphonic music by sparse coding,” *IEEE Trans. Neural Networks*, vol. 17, no. 1, January 2006, pp. 179-196.

[13] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, Wiley, 2000.

[14] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 5, July 2002, pp. 293-302.

[15] N. Scaringella and D. Mlynek, "A mixture of support vector machines for audio classification," *Music Information Retrieval Evaluation Exchange*, http://www.music-ir.org/evaluation/mirex-results/articles/audio_genre/scaringella.pdf, 2005.

[16] M. F. McKinney and J. Breebaart, "Features for audio and music classification," *Intl. Conf. Music Information Retrieval*, http://ismir2003.ismir.net/papers/McKinney.pdf, 2003.

[17] A. Rauber, E. Pampalk, and D. Merkl, "Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity," *Proc. 3rd Intl. Conf. Music Information Retrieval*, Paris, France, 2002, pp. 34-41.

[18] M. Esther, H. P. Kriegel, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proc. 2nd ACM Intl. Conf. Knowledge Discovery and Data Mining*, Portland, Oregon, 1996, pp. 226-231.

[19] Y. E. Kim and B. Whitman, "Singer identification in popular music recordings using voice coding features," *Proc. 3rd Int. Conf. Music Information Retrieval*, Paris, France, 2002, pp. 164-69.

[20] F. Pachet and D. Cazaly, "A taxonomy of musical genres," *Proc. Content-Based Multimedia Information Access (RIA0)*, Paris, France, http://www.csl.sony.fr/downloads/papers/2000/pachet-riao2000.pdf , 2000.

[21] F. Morchen, A. Ultsch, M. Theis, and I. Lonken, "Modeling timbre distance with temporal statistics from polyphonic music," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 14, no. 1, January 2006, pp. 81-91.

[22] M. Chan and J. Potter, "Recognition of musically similar polyphonic music," *Intl. Conf. Pattern Recognition*, vol. 4, August 2006, pp. 809-812.

[23] F. Gouyon, F. Pachet, and O. Delerue, "On the use of zero crossing rate for an application of classification of percussive sounds." *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*, Verona, Italy, 2000, pp. 147-152.

[24] J. Foote and S. Uchihashi. "The beat spectrum: a new approach to rhythm analysis," *IEEE Int. Conf. Multimedia,* 2001, pp. 881-884.

[25] *Gracenote: Music Fans*, http://www.gracenote.com/music, Gracenote, 2006.

[26] *freedb Home*, http://www.freedb.org, MAGIX AG, 2006.

[27] R. Typke, *Melodyhound: Search within the music*, http://www.melodyhound.com, 2006.

[28] *Pandora Internet Radio*, http://www.pandora.com, The Music Genome Project, 2006.

[29] *MusicIP*, http://www.musicip.com, MusicIP Corporation, 2006.

[30] *The Complete MIDI 1.0 Detailed Specification v. 96.1*, MIDI Manufacturers Association, Los Angeles, California, 2001.

[31] *Allmusic*, http://www.allmusic.com, All Media Guide, accessed 2006.

[32] B. Y. Chua and G. Lu, "Improved perceptual tempo Detection of music," *Proc. Intl. Multimedia Modeling Conference*, 2005, pp. 316-21.

[33] T. Kitahara, M. Goto, and H. G. Okuno, "Musical instrument identification based on F0 dependent multivariate normal distribution," *Proc. Intl. Conf. Acoustics, Speech and Signal Processing*, vol. 5, 2003, pp 421-24.

[34] A. Meng, P. Ahrendt, and J. Larsen, "Improving music genre classification by short time feature integration," *Proc. Intl. Conf. Acoustics, Speech and Signal Processing*, vol. 5, 2005, 497-500.

[35] C. Xu, N. C. Maddage, X. Shao, F. Cao, and Q. Tian**,** "Musical genre classification using support vector machines," *Proc. Intl. Conf. Acoustics, Speech, and Signal Processing*, vol. 5, 2003, pp. 429-32.

[36] J. S. Downie, *The Music Information Retrieval and Music Digital Library Evaluation Project*, http://www.music-ir.org/evaluation/mirex-results/audio-genre/index.html, Music Information Evaluation eXchange, 2005.

[37] G. Tzanetakis, *Marsyas - About*, http://opihi.cs.uvic.ca/marsyas, 2006.

[38] American National Standards Institute, *American National Psychoacoustical Terminology*, S3.20, New York, American Standards Association, 1973.

[39] N. Beckmann, H. P. Kreigel, R. Schneider and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," *Proc. ACM Intl. Conf. Management of Data*, Atlantic City, New Jersey, 1990, pp. 322-331.

[40] ISO/IEC 13818-3:1998, Information technology -- Generic coding of moving pictures and associated audio information -- Part 3: Audio, International Organization for Standardization, ISO/IEC 13818-3, Geneva, Switzerland, 1998.

[41] *Hydrogenaudio Forums*, http://www.hydrogenaudio.org/forums/index.php, hydrogenaudio.org.

[42] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P Flannery, *Numeric Recipes in C++: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, 2002.

[43] B. Bos, *Hammerhead Rhythm Station*, http://www.threechords.com/hammerhead/.

[44] S. Molau, M. Pitz, R. Schluter and H. Ney, "Computing Mel-frequency cepstral coefficients on the power spectrum," *Proc. Intl. Conf. Acoustics, Speech and Signal Processing*, vol. 1, May 2001, pp. 73-76.

[45] S. G. Mallat, *A Wavelet Tour of Signal Processing*, New York: Academic, 1999.

[46] D. L. Morton Jr., *Sound Recording: The Life Story of a Technology*, Johns Hopkins University Press, 2006.

Appendix A: Mel Frequency Cepstral Coefficients

The Mel Frequency Cepstral Coefficients (MFCCs) are values that are primarily used in speech recognition as a way to represent speech patterns that is in line with what is known about human pitch perception and the capability of human vocal chords. Their original introduction was in [10], whereas [44] provides an alternative method of extraction. This appendix describes the method of MFCC extraction used internally in Marsyas. The MFCC extraction process begins with a segment of N samples of an audio signal, to which a Hamming window function w_h is applied to accentuate the center of the window:

$$w_h(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2 \cdot \pi \cdot n}{N-1}\right) \tag{A.1}$$

Next, the Fast Fourier Transform (FFT) is applied to this window to create the complex spectrum S. The spectrum is then converted into a power spectrum S_p, eliminating the imaginary part:

$$S_p(m) = re\big(S(m)\big)^2 + im\big(S(m)\big)^2 \tag{A.2}$$

At this point, a series of triangular digital filters are applied to the spectrum to directly compute the energy responses in the frequency bands associated with the Mel scale. As described in the main thesis, the Mel scale is linear below 1 kHz and logarithmic above. To shape the spectrum to correspond to the Mel scale, 13 linear filter and 27 logarithmic filter boundaries used in the Marsyas implementation are created using the following equations for the ith filter of each set. The domain for I is $0 \le i \le 12$ for the first equation and $13 \le i \le 39$ in the second:

$$\begin{aligned} f_{linear}(i) &= 133.3333 + 66.6666 \cdot (i-1) \\ f_{\log}(i) &= 933.333 \cdot (1.011703)^i \end{aligned} \tag{A.3}$$

The resulting filter energies vary according to the Mel scale and are highly correlated. To de-correlate the energies, the Discrete Cosine Transform (DCT), which is often used in data compression, is applied to the logarithm of the energies. The DCT represents a function of discrete points, like the filter energies extracted above, by a series of sinusoids of different frequencies and amplitudes. Operating on the 40 filter energies will result in 40 coefficients. In Marsyas, only the lower 13 are kept, as the higher-order MFCCs are much more dependent on pitch. The general DCT equation computes the *k*th MFCC based upon the filter bank energies of *f*, the set of filters described in Equation A.3:

$$MFCC_k = \sum_{n=0}^{39} f_n \cdot \cos\left(\frac{\pi}{N} \cdot \left(\frac{n+1}{2}\right) \cdot k\right) \tag{A.4}$$

Appendix B: List of Test Recordings

Group 0: Cui preludes for piano
Cesár Cui - Preludes Op. 64
All taken from Naxos 555557

Group 1: Chopin preludes for piano
Fryderyk Chopin - Preludes Op. 28, Op 45
All taken from Naxos 554536

Group 2: Romantic piano music
Amy Beach – Waltz-Caprice Op. 4
Taken from Koch International Classics 37254
Anton Bruckner – Betrachtung, Quadrille Mvt. 1
All taken from CPO 999256
Fryderyk Chopin – Berceuse Op. 57, Mazurka Op. 17-4, Prelude Op 28-20
Taken from Naxos 550358, Naxos 554527, Naxos 554536
Cesár Cui – Prelude Op. 64-23
Taken from Naxos 555557
Sergey Rachmaninov – Etudes-Tableaux Op 39-1, Op. 39-8
Taken from EMI 85817
Franz Schubert - Marche D602 No. 2
Taken from EMI 69764
Arthur Scriabin – Impromptu Op 12-1
Taken from Vox 3606
Leo Tolstoy – Waltz
Taken from Bis 1502

Group 3: Bach Golberg variations for piano
Johann Sebastian Bach – Goldberg Variations
All taken from CBS Recordings B0000025PM

Group 4: Avant-garde/atonal/etc. piano music
Luciano Berio – Sequenza No. 4
Taken from Naxos 8557661
Pierre Boulez – Sonata No. 2 Mvt. 3, Sonata No. 3 Mvt. 3
Both taken from Montaigne 061
John Cage – 34'46.776 extract, Music of Changes, Music for Piano No. 1, Winter Music
Taken from MDG 613 0787-2, Lovely Music LCD 2053, MDG 613 0784-2, MDG 613 0787-2
Morton Feldman – Pieces 1954, 1955, 1959 and 1964
Taken from Mode 53, Sub Rosa 018-41, Sub Rosa 189, Etcetera 300
Arnold Schoenberg – Pieces Op. 23 No. 1
Taken from Deutsche Grammophon 423249

Karlheinz Stockhausen – Mantra Pt. 4
Taken from New Albion 025
Anton Webern – Variations Op 27 No. 1
Taken from Sony Classical 45845

Group 5: Bach cello suites
Johann Sebastian Bach – Cello Suites
All taken from Teldarc 25704 (Bach 2000 Complete Edition)

Group 6: Paganini violin caprices
Niccólo Paganini: Caprices for Violin
Taken from Deutsche Grammophon 469235 (1, 6, 9, 17), EMI 67257 (2, 3, 5, 8, 16), Telarc Digital 80398 (7)

Appendix C: Notes on Results Files

Available along with this document are two spreadsheets in a tab-delimited format that can be imported into the program of the reader's choice. These files each contain result data for the timbre and rhythm experiments as follows:

- `results_timbre.txt`: Results for each of the file groups are placed together. File names are indicated in the far left column; these may be associated with the list of recordings in Appendix B. Each file has its own row of result data. Below each group, averages and standard deviations are shown.

- `results_rhythm.txt`: The values used to generate the histograms are grouped into two columns per histogram. The topmost entry identifies the associated work and the point within the work the histogram refers to, if it is a running histogram.

The files can be used directly by anyone wishing to repeat the analysis given in this thesis. For any further assistance in using them, please contact the author.

Vita

An Alabama native, Zachary W. Bond earned his Bachelor of Science degree from Case Western Reserve University in 2004. His research interests in the field include intelligent systems, pattern recognition, and embedded systems. Zac enjoys applying his computer interest with knowledge in other areas such as music and history.