國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

複音音樂的樂器編制分析及其在音樂相似度估計上的應用

Instrumentation Analysis of Polyphonic Music and Its Application to Music Similarity Measure

許年德

Nien-Teh Hsu

指導教授：貝蘇章 博士

Advisor: Soo-Chang Pei, Ph.D.

中華民國 98 年 6 月

June, 2009

國立臺灣大學（碩）博士學位論文
口試委員會審定書

複音音樂的樂器編制分析及其在音樂相似度估計上的應用
Instrumentation Analysis of Polyphonic Music and Its Application to Music Similarity Measure

本論文係許年德君（學號 R96942047）在國立臺灣大學電信工程學研究所完成之碩士學位論文，於民國 98 年 5 月 23 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

貝蘇亨　（簽名）

（指導教授）

王鵬華　鄭士康

徐忠枝

系主任、所長　王暉　（簽名）

誌謝

能完成這篇論文，首先要特別感謝我的指導教授貝蘇章老師，老師在研究上的熱忱以及經驗一直是我敬佩以及學習的榜樣。並且感謝老師當初願意指導我進行音樂訊號處理相關的研究，讓我達成了一直以來想要嘗試這塊領域的目標。同時也要感謝我的父母親，一路栽培我到進入研究所，使我能夠衣食無缺，放心地專注在課業上，如此的成果理當歸功於你們。

謝謝實驗室一起奮鬥的夥伴們：家宏、仕昕、佩君和 Joost。這兩年來除了在課業上的彼此討論之外，還有發生在實驗室裡的大小有趣事情，現在回想起來也都還歷歷在目。感謝大家一路走來的包容和支持。在此要特別感謝仕昕，在我遭遇挫折或是需要幫忙的時候，你總是義不容辭的伸出援手和表達關心，和你在課業以外的興趣交流也讓我的研究生活不再枯燥乏味。

感謝實驗室的學長姐們，你們認真嚴謹的研究態度激勵了我，讓我時常提醒自己要好好利用身處在學術界的這兩年時間。感謝實驗室的學妹們：二馬、嘟嘟和鳥蛋，你們為整個實驗室帶來了活力和歡笑，謝謝你們一直以來的支持和關心。

最後要感謝正在閱讀這份論文的你，謝謝你從千萬本中論文拿起此篇，這篇論文雖然經過校閱，但仍多有疏漏，請多多包涵與指教。

中文摘要

在過去的數十年裡，由於網際網路的蓬勃發展，各式各樣多媒體檔案的數量不斷增加。在這之中，不論是在獲取或是發佈數位音樂檔案都變得比過去容易很多。也由於此數量規模的不斷爆增，我們需要一個新的聆聽音樂和發掘新音樂的方法。

在這篇論文的一開始，我們會介紹一個簡單的音樂相似度估計系統，並且模擬它的效能。根據實驗結果顯示，使用較低階的特徵向量來描述曲子的特性，並不足以讓我們分離出不同音樂內容本身對相似度造成的影響，例如和絃、曲風、樂器編制和旋律。因此，這篇論文的主要目標在於將原本的低階特徵替換為與音樂內容有關的中階特徵。於此，我們將特別著重於樂器編制的自動化分析。音樂訊號音色的時頻分析和單一樂器的分類問題都將在此篇論文中討論，以做為基本的工具。之後我們將延伸此想法到處理更複雜的複音音樂，並且截取其隨時間變化的樂器編制資訊。藉由在相似度估計系統上使用此資訊，我們發現計算出的相似樂曲結果中，將可以特別針對樂器和音色，而非其他音樂內容。如此將可以取代原本的相似度估計系統，達到實現多模式音樂相似度估計的目標。

關鍵字：基於內容的音樂資訊擷取、樂器分類、音樂相似度估計、音樂訊號處理

Abstract

During the past few decades, the world has ushered in a new era, with booming Internet technology and immense multimedia content distribution. The acquisition and circulation of digital music file become much easier than ever. Due to this rapidly rising of music quantity, a brand new way of discovering and recommending music is thus highly expected.

In the beginning of this study, a conventional music similarity measure system based on the signal analysis methods is implemented and evaluated. According to the experimental results, it shows that the low-level features from signal analysis techniques are not strong enough to fulfill the discrimination between various musical content, such as the chord progression, genre, instrumentation, and melody. Therefore, the aim of this study is to incorporate the low-level feature with the mid-level feature, in order to utilize the musical content. We focus on the way to extract the instrumentation information leaved by the composers. The time-frequency analysis of musical instrumental signals and the classification problem of various instruments in the monophonic case are studied. After that, we extend the idea to deal with the polyphonic music and analyze its time-varying instrumentation information. By incorporating this information back to the original similarity measure system, the calculated similar songs can resemble to each other specifically in the sense of the instrumentation.

Index Term — **Content-based music information retrieval, Instrument classification, Music similarity measure, Audio signal processing**

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Since that music is readily accessible in the compressed digital form [1], recently people were used to store and listen to the music files in their electronic devices, such as the personal computers and MP3 players. This trend leads to a significant increase in the personal music collection size: it can easily exceed the practical limit on the time we have to listen to them. For instance, one thousand music tracks inside a personal music device contain a total duration of approximately three days of nonstop continuous audio [2]. The compression technique and the rising of Internet technology also make the distribution of new music recordings become much easier. The sales volume of traditional CD stores has rapidly deteriorated due to the fact that online music stores gradually seize the market [3].

Conventional ways of listening and discovering music, such as listening to the radio broadcasting and buying the CDs from the record stores, are no longer sufficient for the immense music pieces in our life. On the contrary, a personalized

and automatic recommendation system is thus required. So far until now, the most common method of accessing the music content is through the textual metadata. A well-known example of the online metadata-driven music recommendation service is *Pandora.com*[1]. The site is originally created by the Music Genome Project [4]. It maintains a large database which contains a set of human-entered metadata, such like the artist, genre, and recording year. Users enter a song or artist that they enjoy, and the service can automatically respond to play selections that are musically similar in the sense of the metadata. The other example of the commercial systems is *Last.fm*[2], a music community website, which has similar functionalities as *Pandora.com*. The drawback of these systems, however, is they require great time for preparing the database that contains all information needed. For instance, a company may need to hire a team to enter these information before starting the recommendation process. When the size of the database become large, the cost also grows. Hence it cannot be generalized to large-scale music databases.

To address this problem, we introduce the technique of automatic music similarity measure, as part of content-based music information retrieval task, aiming at using a computer-based method to compute the similarity between music tracks by means of the audio signal processing methods. To date, several schemes have been proposed by researchers. The methods can be roughly characterized to three main stages: the feature extraction step, cluster modeling step, and distance calculation step. To our best knowledge, most of the existing systems applied the low-level feature set, such like the Mel-frequency cepstral coefficients, in their first stage. Nevertheless, the idea of using these low-level features is usually taken from

[1]http://www.pandora.com

[2]http://www.last.fm

the speech or audio processing applications. Consequently, it cannot make use of the musical content. The main purpose of this study is to propose a novel similarity measure system with the low-level feature replaced by the content-related mid-level features. This purpose is identified as being of importance to implement a real-world music recommendation system that can automatically generate a playlist in every different aspects [5] (i.e., toward the multi-modal music recommendation system).

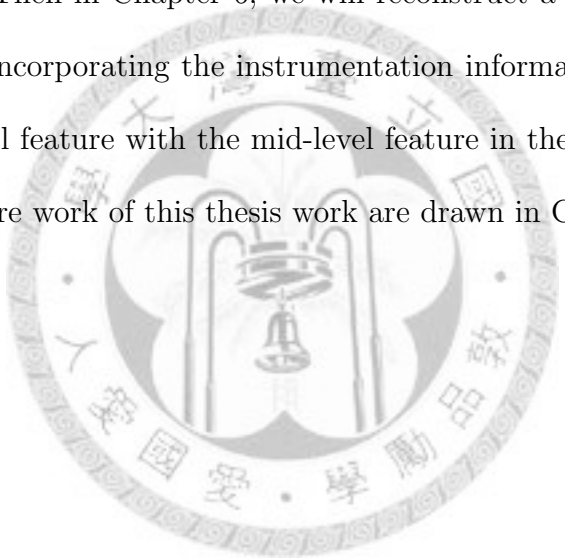
1.2 Primary Achievements of This Study

There are two main notable achievements in this study. The first one is to design an efficient instrumentation analysis algorithm for dealing with the polyphonic music. In contrast with the existing algorithms, it can give an additional time-varying information about the dominance of each instrument. This part of the work is illustrated in Chapter 5 and published in [6]. The second one is to further apply this instrumentation information as a mid-level feature to design a novel music similarity measure system. This part of the work is introduced in Chapter 6 and also published in [7].

1.3 Organization

The rest of this study is structured as follows. In Chapter 2 we discuss several basic ideas inside the music similarity measure systems. A simple and efficient system with low-level feature extraction is implemented and evaluated. The shortness and insufficient parts of the method will also be emphasized and summarized. In Chapter 3, we focus on how to use the constant Q transform to analyze an instrumental signal

by its time-frequency distribution. In Chapter 4, we deal with the simplest task in the instrument recognition task – the isolated notes of the monophonic music pieces. Different approaches are compared and ranked in terms of the recognition rates to the benchmark database. Additionally, in Chapter 5, we move on to tackle the recognition problem in the polyphonic case. In contrast to previous studies, the aim of our work is trying to extract the time-varying instrumentation information that leaved by composers, in the musicology point of view. A beat-synchronous feature integration process in combination with the fuzzy clustering technique is applied to solve this problem. Then in Chapter 6, we will reconstruct a new music similarity measure system by incorporating the instrumentation information. That is to say, combine the low-level feature with the mid-level feature in the framework. Finally, conclusions and future work of this thesis work are drawn in Chapter 7.



Chapter 2

A Music Similarity Measure System

2.1 Introduction and Related Work

Music similarity measure has become a big issue among the area of audio signal processing. At the present there is more and more music files stored on personal computers, in music libraries, or even on Internet. In order to search and choose the desired music by user in an efficient way, some features need to be extracted from the music files for users to make their own choices. Although traditional approaches that dealing with the embedded metadata (e.g., artist, performer, title and genre) were presented, these human-generated tags suffer from their limited applicability in many music-related applications [8].

Moreover, by the help of quick-growing computation speed of digital signal processing, it becomes much easier for people to analyze and synthesize audio files by the computer. Hence automatic music information retrieval is considered as an ef-

ficient and essential technique nowadays comparing to the traditional approaches [9]. The purpose of designing a music similarity measure system is to implement a framework, which allows to compute the distance of each song in pair. Here the term *distance* is regarded as the perceptual dissimilarity between two music pieces. The similarity could lie in many different musical aspects, such as same genre, same artist, or same album. Therefore, the evaluation process of similarity measure systems could also differ and depends on the purpose of the application.

So far, several systems have been introduced and implemented by researchers. Logan and Saloman presented a method to compare songs based on the traditional signal analysis technique [10]. In their system, each song is identified as a unique signature, which is based on the k-means clustering algorithm of the spectral feature. Then the signatures are compared by the earth mover's distance [11]. Pampalk *et al.* made some modification for reducing the computation time, and also some other improvements [12], [13]. These works also participated in the International Society for Music Information Retrieval (ISMIR) contest. The results are evaluated in terms of the genre classification and the artist identification tasks. Pampalk *et al.* proposed a method to efficiently access and explore the music pieces by similar signal processing method [14], [15]. The output result is called *Island of Music*. It uses a 2-D plane to display the similarities between songs on several isolated regions. In addition, Aucouturier and Pachet described the usage of music similarity measure in [16]. The framework of their system is similar to the previous ones, but with different clustering and distance measure methods selection. They evaluated the result by both objective and subjective criteria. Aucouturier *et al.* also made a comprehensive study on the parameter selection problems of each stage [17], [18]. In their work, they especially emphasized on the calculation of the timbre similarity.

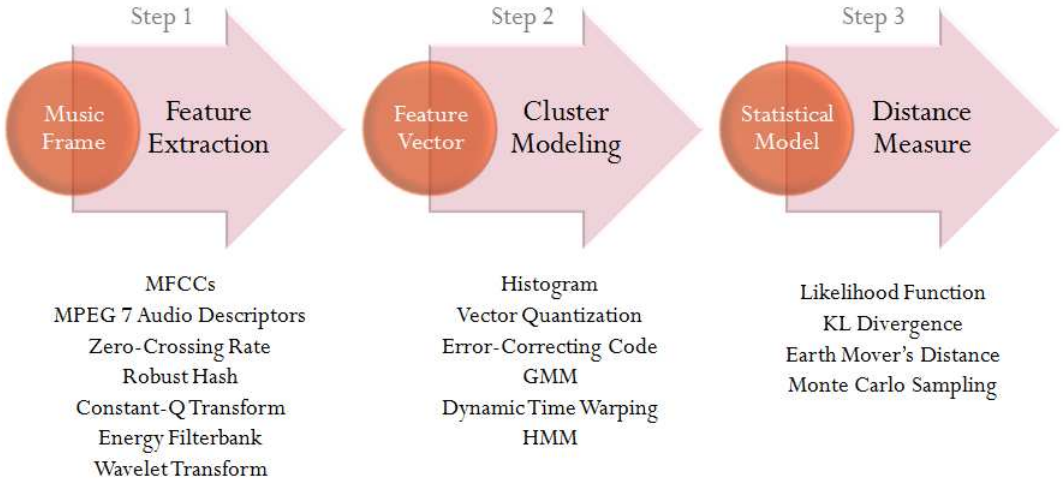


Figure 2.1: A basic framework of the music similarity measure system. Some of the possible approaches in each stage are also listed.

While music similarity measure technique gains more interest from researchers, at the same time several development toolkits are published. The first one is *Marsyas* [19], an open source software framework for audio processing with specifically emphasis on the music information retrieval applications. *MA Toolbox* [20] and *DTU Toolbox* [21] also play an important role. The major difference is that they are developed under Matlab environment. These applications implement the calculation process discussed before and with slightly different feature selections. However, one can easily find that their performance is highly dependent on the situation of execution.

Among these research works, most of the systems can be roughly characterized to a three-stage pattern recognition framework. The corresponding three stages are as follows: the feature extraction step, the cluster modeling step, and the distance measure step. Figure 2.1 shows a basic framework of similarity measure system and some of their possible approaches. We will introduce the detail of each stage in the

following sections.

2.2 Feature Extraction

The aim of extracting the feature vector from a music piece is to obtain a meaningful and uniform vector to represent the music itself in the desired manner. There are several reasons that we cannot compare the songs directly by their amplitude. First of all, in psychoacoustics, human usually "listens" in the frequency domain instead of time domain. Secondly, the similarity in perception usually lies within the high-level characteristics, like genre and tempo. Last but not least, if we do so, a simple time-shift applying to the music piece will lead to a completely different calculation result. Therefore, a generalized feature extraction algorithm is required for further processing.

Since the development of audio signal processing is much later than that of speech signal processing area, many ideas come from the field of speech processing. One of the most representative instances is the Mel-frequency cepstral coefficients (MFCCs). In MFCCs, the frequency bands are positioned logarithmically, which approximates the response of the human auditory system. It performs better than the linear-spaced frequency bands, such like the discrete Fourier transform (DFT) and the discrete cosine transform (DCT). In addition, there are still many other features that are widely used in this area. Spectral centroid, spectral roll-off, time domain zero-crossing rate together form a solid combination of the feature vector. Followings are the detail description of each feature set.

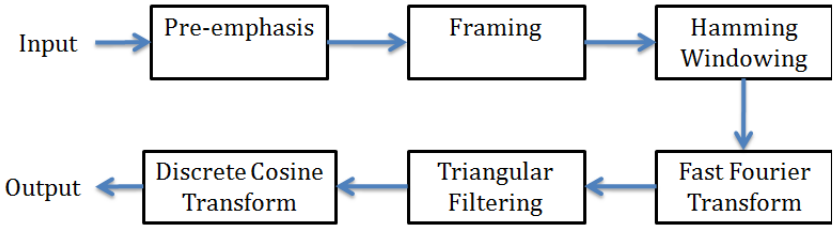


Figure 2.2: Block diagram for computing the MFCCs.

2.2.1 Mel-Frequency Cepstral Coefficients

The Mel-frequency cepstral coefficients (MFCCs) is a representation defined as the real cepstrum of a windowed short-time signal derived from the fast Fourier transform (FFT) of that signal [22]. It is considered as the most widely-used spectral feature in audio and speech signal processing applications.

Figure 2.2 shows the block diagram for computing the MFCCs. First of all, the audio signal is sent to a high-pass filter:

$$s_{\text{out}}[n] = s_{\text{in}}[n] - a \cdot s_{\text{in}}[n-1], \tag{2.1}$$

where $s_{\text{in}}[n]$ and $s_{\text{out}}[n]$ denote the input and output signal, respectively. The value of a is usually selected between 0.9 and 1. The goal of the pre-emphasis process is to compensate the high-frequency component that was suppressed during the sound production mechanism. This phenomenon is especially obvious for the human vocal tract model.

Next, the input audio signal is then blocked into frames of approximately 20 to 30 ms with overlap of 1/3 to 1/2 of the frame size. This is due to the non-stationary property of the audio signal. Thus we need to segment the signal to considerably small pieces to make it close to stationary. Usually the frame size is set to equal to power of two in order to facilitate the use of the FFT algorithm. If it is not the

case, the zero-padding process needs to be applied.

After that, we multiply each frame with a Hamming window in order to improve the continuity of the beginning and the ending points inside the frame. Let $s[n]$, $n = 0, ..., N-1$ denote the signal in a frame, then the signal after applying Hamming windowing is $s[n] \cdot w[n]$, where $w[n]$ is the Hamming window defined by

$$w[n, \alpha] = (1 - \alpha) - \alpha \cos[\frac{2\pi n}{N - 1}] \tag{2.2}$$

In the fourth step, we perform the FFT to obtain the magnitude frequency response of each frame. Spectral analysis shows that different timbres in audio signals correspond to different energy distribution over the frequency bands. Then, the most crucial step should be the triangular band-pass filtering. We multiply the FFT result by a set of twenty triangular band-pass filters to get the log energy of each filter. The formula of the filter bank is

$$B_m(k) = \begin{cases} 0, & \text{if } k < f_{m-1} \\ \frac{k - f_{m-1}}{f_m - f_{m-1}}, & \text{if } f_{m-1} \le k \le f_m \\ \frac{f_{m+1} - k}{f_{m+1} - f_m}, & \text{if } f_m \le k \le f_{m+1} \\ 0, & \text{if } f_{m+1} < k \end{cases} \tag{2.3}$$

This process is designed to imitate the functionality of the human auditory system. Figure 2.3 shows a visualization of the triangular filter bank. This figure is plotted using the *Audio Processing Toolbox* developed by Jang [23].

In the final step, we apply the discrete cosine transform (DCT) on the twenty log energies E_k, $k = 1, ..., 20$ obtained from the triangular band-pass filtering to derive the L Mel-scale cepstral coefficients. Here we use the type-2 DCT as follows:

$$C_m = \sum_{k=0}^{N-1} E_k \cos[\frac{\pi}{N}(k + \frac{1}{2})m] \tag{2.4}$$

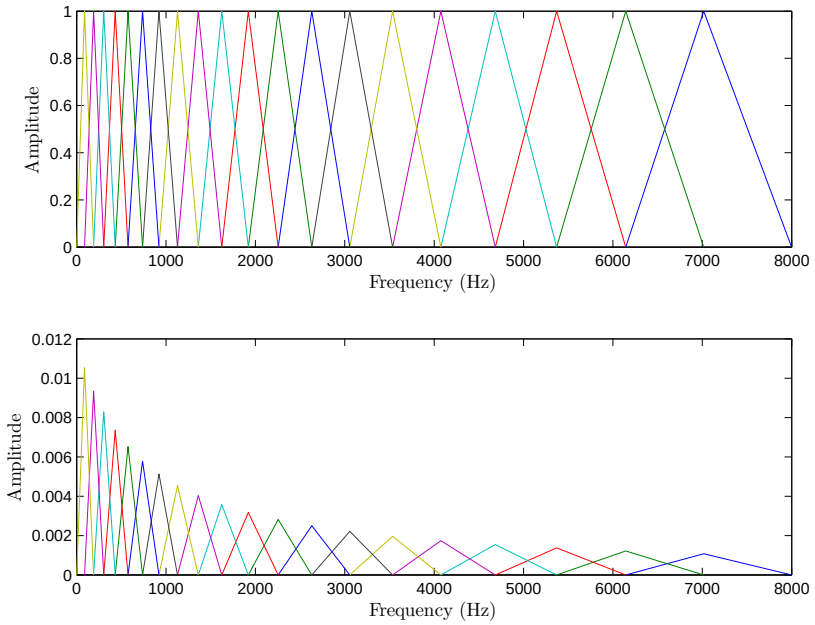


Figure 2.3: Triangular filter bank used in the computation of the MFCCs. The top plot shows the original triangular filter bank. The bottom plot is the same but with normalization.

Since we have already performed the FFT algorithm, the DCT transforms the frequency domain into a time-like domain called the *quefrency domain*. The obtained features are similar to the cepstrum. The purpose is to further reduce the feature size, and in practice we only consider the 1st to 13th vectors of the DCT result.

2.2.2 Timbral Texture Feature

Except for the MFCCs, there are also several other timbre features suggested by researchers. Here we introduce some of the spectral features suggested in [9]. These features are based on the short-time Fourier transform (STFT) computation and are calculated for every single frame as well. Followings are the description of these features:

1. **Spectral Centroid**:

   The spectral centroid of a frame is defined by the gravity center of the mag-

nitude spectrum of the STFT,

$$G_t = \frac{\sum_{n=1}^{N} M_t[n] \times n}{\sum_{n=1}^{N} M_t[n]} \tag{2.5}$$

It represents a rough sketch of the spectral shape. The frame with higher spectral centroid usually leads to higher frequency combination.

2. **Spectral Roll-off**:

   The spectral roll-off is defined as the frequency R_t below which 85% of the magnitude distribution is concentrated,

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 \times \sum_{n=1}^{N} M_t[n] \tag{2.6}$$

   It can be regarded as another rough representation of the spectral shape.

3. **Spectral Flux**:

   The spectral flux is defined as the squared difference between the normalized magnitude of successive spectral distributions,

$$F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2 \tag{2.7}$$

   The spectral flux is a measure of the total amount of the spectral change.

4. **Time-Domain Zero Crossing Rate**:

   The time-domain zero crossing rate is defined as

$$Z_t = \frac{1}{2} \sum_{n=1}^{N} |\text{sign}(x[n]) - \text{sign}(x[n-1])| \tag{2.8}$$

   It provides a measure of the noisiness of the signal, and could also be used in the separation process of the audio and speech signal.

Table 2.1: Summary of the parts included in the MPEG-7 standards.

MPEG-7 Systems	Tools for efficient transport and storage
MPEG-7 DDL	For defining the syntax
MPEG-7 Visual	Dealing with visual descriptions
MPEG-7 Audio	Dealing with audio descriptions
MPEG-7 Multimedia Description	For generic features and multimedia descriptions
MPEG-7 Reference Software	Software implementation of relevant parts
MPEG-7 Conformance Testing	For testing the conformance of implementations
MPEG-7 Extraction and usage	Informative material about the usage
MPEG-7 Profiles and levels	Provides guidelines and standard profiles
MPEG-7 Schema Definition	Specifies the schema using the DDL

2.2.3 MPEG-7 Audio Descriptors

MPEG-7 is an ISO/IEC standard developed by the Moving Picture Experts Group (MPEG). This committee has also developed several multimedia standards: MPEG-1, MPEG-2 and MPEG-4, which are widely used in the compression and transmission applications. MPEG-7, formally named as *Multimedia Content Description Interface*, is a standard aiming at describing the the content data of multimedia. It supports the exposition of the information meaning, which can be accessed by a device or a computer code. MPEG-7 does not aim at any particular application; instead, the elements inside MPEG-7 standards support a wide range of applications as possible [24], [25].

Table 2.1 lists a summary of the parts included in MPEG-7. Although the standards contain rich material, in this work only the MPEG-7 Audio Descriptors are used. These descriptors are going to be applied as the basic low-level features for implementing the instrumentation analysis system (see Section 5.3.1 for more information).

2.3 Cluster Modeling

After the feature extraction step is done, the modeling block usually receives a sequence of feature vectors calculated on a frame-by-frame manner. The number of frames for a music piece mainly depends on the duration. Exploiting redundancies in the frame time vicinity inside a music piece is useful for further reduce the feature size. By clustering the feature vectors, it can help us to derive a statistical model of the music characteristics. There are several possible methods that can be chosen from. The methods can be mainly divided into two classes: the static approach and the dynamic approach.

Histogram is one of the simplest example of the static method. It mixes all the time frames together, and statistically calculates the counts correspond to every possible value. Vector quantization (VQ), Gaussian mixture models (GMMs), and k-means algorithm also belong to this class. Notice that these approaches cannot reveal the time property, since they disarrange the order of the time sequence. For another, the hidden Markov models (HMMs) is considered as a well-known example of the dynamic approach. It consists of several hidden states and uses the transition probabilities to describe the characteristics of the feature vectors.

One should believe that in most cases the dynamic approach would outperform the static approach, since the former one utilizes the time information. But in [26], the authors argue that in the polyphonic situation, the above argument is not always true. The reason is that the dynamic nature of the polyphonic timbre frames are difficult to model correctly. Following is an introduction to two famous clustering methods: the k-means clustering algorithm and the GMMs.

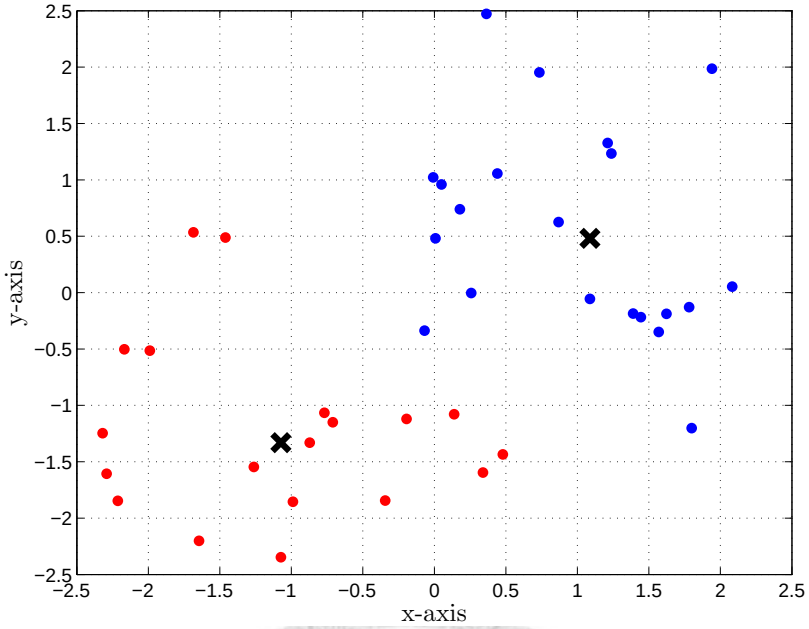


Figure 2.4: A 2-D demonstration of the k-means algorithm.

2.3.1 k-means Clustering

The k-means clustering methods, as one of the simplest unsupervised learning algorithms, was original proposed by MacQueen in 1967 [27]. The main idea is to derive k centroids, as the representative of each cluster, to minimize total intra-cluster variance. The squared-error objective function is

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{2.9}$$

where S_i denotes the ith cluster set, $i = 1, ..., k$ and μ_i is the centroid of all the points inside S_i. The algorithm is composed by two iterative phases. First of all, the centroids should be placed in a scheming way because that different locations will cause different results. The better choice is to place them as much as possible far away from each other. The next step is to make each point belong to a given data set and associate it to the nearest centroid. These two steps are repeated until there is no further improvement. Table 2.2 lists the detailed algorithm of the k-means

Table 2.2: Detailed algorithm of the k-means clustering.

Step 1:	Choose k points as initial centroids.
Step 2:	Assign each data to the closest centroid.
Step 3:	Recalculate the centroids according to the above assignment.
Step 4:	Repeat 2 and 3 until the objective function is no longer improved.

clustering.

Figure 2.4 shows a 2-D executing result of the algorithm. The cross signs denote the cluster centroids. The points belong to two clusters are marked as red and blue to be clearly identified. In addition, there are several considerations in the clustering algorithm. First, the way to initialize the centroids is not specified, but it tends to make an influence on the clustering result. One popular approach is to do the random selection, repeat several times to inspect the results and choose the best one. Secondly, the result also depends on the selection of the distance metric. In (2.9), we adopt the simplest one – the Euclidean distance in the calculation. However, there are also other possible choices, such like the city block distance and the cosine metric. For the last one, comparing to other hierarchical clustering methods, choosing the proper value of k in the algorithm is a challenging task.

Consider our application in the music similarity measure task, if there is T_1 feature vectors correspond to T_1 frames that are extracted from a music file S_1, we then apply the clustering algorithm to reduce the data size to only k central vectors. That is to say, for every song S_i with variable frame size T_i inside the music database, we unify theses vectors to k centroids in order to make a comparison of them.

2.3.2 Gaussian Mixture Models

Unlike the non-parametric k-means algorithm, the Gaussian mixture models (GMMs) can be regarded as a model-based clustering method. It assumes the data under modeling is generated via a probability density function (PDF), which is the weighted sum of a set of Gaussian PDFs. In practice, with a proper mixture number selection, one can use the GMMs to fit any kind of distributions. Moreover, by using the expectation-and-maximization (EM) algorithm, we can identify the optimal parameters for the GMMs in an iterative manner. The detailed algorithm of EM can be found in [28].

A finite mixture model is a distribution of the form

$$p(\mathbf{x}) = \sum_{j=1}^{g} \pi_j p(\mathbf{x}; \mu_j, \mathbf{\Sigma}_j) \tag{2.10}$$

where g is the number of mixtures, π_j represents the *partial membership* subjected to

$$\sum_{j=1}^{g} \pi_j = 1 \tag{2.11}$$

We use μ_j and $\mathbf{\Sigma}_j$ to denote the mean and covariance matrix of the jth mixture, respectively. With a initial guess for the parameters of the mixture model, the partial membership of each data point in the elemental distribution can be computed by calculating the expectation values for the membership variables of each data point. This is considered as the *expectation step*. The probability that $\mathbf{x}_i$ belongs to mixture j given the current estimates is

$$\omega_{i,j} = \frac{\pi_j p(\mathbf{x}_i; \mu_j, \mathbf{\Sigma}_j)}{\sum_{k=1}^{g} \pi_k p(\mathbf{x}_i; \mu_k, \mathbf{\Sigma}_k)} \tag{2.12}$$

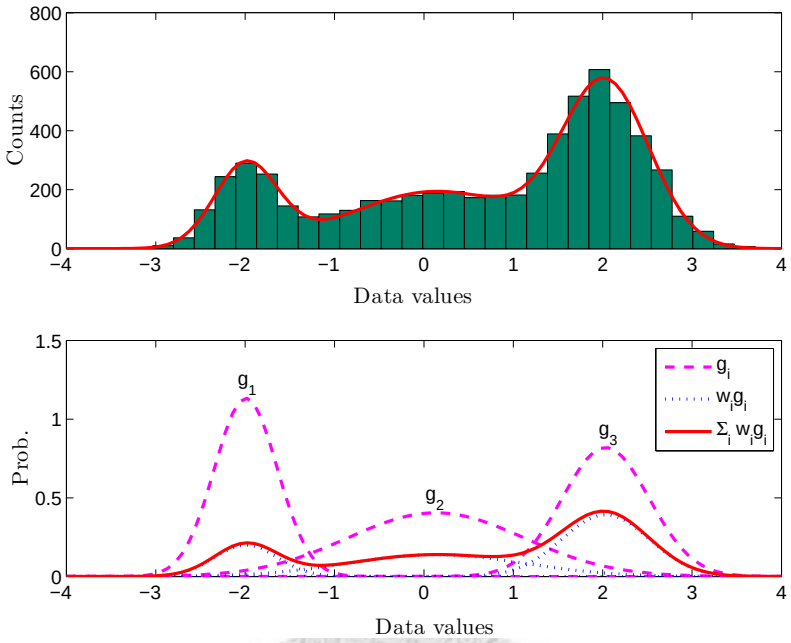


Figure 2.5: A 1-D demonstration of GMMs. The top plot shows the data histogram. The bottom plot shows the GMM calculation results with three mixtures.

Consider the *maximization step*, we re-estimate μ_j and $\mathbf{\Sigma}_j$ by

$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} \omega_{i,j}\mathbf{x}_i}{\sum_{i=1}^{n} \omega_{i,j}} \tag{2.13}$$

and

$$\hat{\mathbf{\Sigma}}_j = \frac{\sum_{i=1}^{n} \omega_{i,j}(\mathbf{x}_i - \hat{\mu}_j)(\mathbf{x}_i - \hat{\mu}_j)^T}{\sum_{i=1}^{n} \omega_{i,j}} \tag{2.14}$$

Figure 2.5 shows a 1-D demonstration of the GMMs calculation. This demo is plotted using the *DCPR Toolbox* developed by Jang [29]. In this example three mixtures are applied to approximate the distribution. Comparing to the k-means algorithm, not only the means (related to the centroid in k-means) but also the covariance matrix is computed and stored in the GMMs. Therefore, for a proper choice of mixture number g, one can expect the GMMs should outperform the k-means algorithm for obtaining a precise statistical information.

2.4 Distance Measure

After receiving a set of the statistical models, say the centroids from the k-means or the means and covariance matrices from the GMMs, the remaining question is as follows: how can we estimate the pairwise distance between these model?

In statistics, calculating the likelihood function of two models should be one of the most straight forward approaches. Nevertheless, there are still many other approaches have been proposed as well. For example, the Kullback-Leibler divergence (KLD) aims at measuring the difference between two PDFs, and it is also correlated to other quantities in information theory [30], [31]. Robner suggested another dissimilarity measure called the earth mover's distance (EMD) that is originally designed for the image retrieval works [11]. It is also possible to just simply sample from one statistical model and compute the likelihood of the samples given the other GMMs [16]. We will go into the detail of these approaches in the following section.

2.4.1 Likelihood Function

In statistics, a likelihood function is a conditional probability function regarded as a function of its second argument B and with its first argument A held fixed

$$b \mapsto P(A|B = b) \tag{2.15}$$

In our application, the likelihood function is used to calculate the probability of the feature vector, say A's MFCCs, given the statistical model, say B's GMMs. Being a straightforward and reasonable approach; however, it requires to access A's MFCC data, which is not desired to be stored in the database (i.e., only the models are desired to be stored). It also requires large computational time for computing all the possible MFCC vectors.

2.4.2 Kullback-Leibler Divergence

In probability and information theory, the Kullback-Leibler divergence (KLD) is a non-commutative measure of the difference between two probability distributions [32]. For the probability distributions P and Q as discrete random variables, the KLD of Q from P is defined as:

$$\mathrm{KLD}[P||Q] = \sum_i P[i] \log \frac{P[i]}{Q[i]} \tag{2.16}$$

On the contrary, in continuous cases it is defined as:

$$\mathrm{KLD}(P||Q) = \int_{-\infty}^{\infty} P(x) \log \frac{P(x)}{Q(x)} dx \tag{2.17}$$

Although the KLD is often regarded as a distance measure, actually it is not a real metric due to its asymmetric property. From the information theory point of view, the KLD measures how inefficient on average it would take to code one distribution using the other one as the codebook. However, the drawback is that it only considers the dissimilarity bin-by-bin instead of cross-bin.

2.4.3 Earth Mover's Distance

The earth mover's distance (EMD) [11] is suggested to overcome the drawback from the KLD: it takes the additional cross-bin information into account. Given two distributions, one can be treated as a mass of earth properly spread in space, the other as a collection of hole in that the same space. This is the physical meaning of the EMD. Computing the EMD is based on a solution to the well-known *transportation problem*. Suppose that there is a set of suppliers, each with a given amount of goods, are required to supply several consumers, each with a given limited capacity. The

algorithm aims at finding a least amount of work, on the flow of goods form the suppliers to the consumers that best satisfies consumer's demands.

Let $P = \{(p_1, w_{p_1}), ..., (p_m, w_{p_m})\}$ denote the first signature with m clusters, where p_i represents the cluster representation and w_p is the weight of the cluster. $Q = \{(q_1, w_{q_1}), ..., (q_n, w_{q_n})\}$ denotes the second signature with n clusters, and $D = [d_{ij}]$ denotes the ground-truth distance matrix where d_{ij} is the ground-truth distance between clusters p_i and q_i.

The goal is to find a flow $F = [f_{ij}]$, with f_{ij} being the flow between p_i and q_j, that minimizes the overall cost

$$\text{WORK}(P, Q, F) = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij} \tag{2.18}$$

subject to the following constraints:

$$f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{p_i} \qquad 1 \leq i \leq m$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{q_i} \qquad 1 \leq j \leq n$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = \min(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j})$$

Once the transportation problem has been solved, then we have found the optimal flow F, the EMD is defined as the work normalized by the total flow:

$$\text{EMD}(P, Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}} \tag{2.19}$$

2.4.4 Monte-Carlo Sampling

The Monte Carlo sampling method was suggested by Aucouturier in his music similarity application [16], [17], [18]. It can be regarded as a simplification of the likelihood calculation. The method begins with a sampling process from one statistical

model, and to compute the likelihood of the samples given the other model. This process could roughly corresponds to recreating a song from its timbre model, and applying the likelihood method defined above to this newly created song and the model of the other song. To be more specific, we sample a large number of points S^A from model A and compute the likelihood of the samples with given model B. The formula is as follows:

$$D(A,B) = \sum_{i=1}^{NS} \log P(S_i^A/A) + \sum_{i=1}^{NS} \log P(S_i^B/B) - \sum_{i=1}^{NS} \log P(S_i^A/B) - \sum_{i=1}^{NS} \log P(S_i^B/A) \quad (2.20)$$

where NS is the number of samples drawn from each distribution. We normalize the distance to lie in 0 and 1 and make them symmetric.

This approach is well suited for large scale musical databases, because we do not need to store the feature vectors, but only the parameters from statistical models. The authors also show that in experiment when NS is large enough, the result should approach to the likelihood calculation.

2.5 Simulation Results

The experiment and evaluation is developed by the *Music Similarity Toolbox* (MA Toolbox)[1] [20] implemented by Pampalk. It consists of three stages, as described in the previous sections. Here we construct a small music database with thirty songs, covering a variety of different genres. The distances between each song in pair are calculated and normalized, in order to make a comparison. We will also analyze and verify the simulation results in the end of this section.

[1]http://www.ofai.at/ elias.pampalk/ma/

2.5.1 Music Similarity Measure Toolbox

The Music Similarity Measure Toolbox (MA Toolbox) is a collection of functions designed for Matlab environment. It contains several functions to analyze the music files and compute their similarities. The toolbox implements several functions to visualize intermediate steps in the computations. Furthermore, some extra functionalities are also included to create the *Island of Music*, where islands represent the clusters of similar pieces [14]. Although the toolbox is not suitable for dealing with large scale databases, it can help us to take a glance at the implementation aspect of similarity measure process.

2.5.2 Experiment Setup

The database consists of thirty songs with different genres representation as their tags. A wide range of different genres are collected (e.g., Jazz, Classical, Popular, Rock, Metal, and Dance). Table 2.3 lists the title, artist, and the related genres for each song.

To ensure the variety of different recording qualities, the pieces were taken from radio, CD, and MP3 compressed audio files. The files were all stored as 11025Hz, 16-bit, and mono audio format for computational efficiency. In this experiment, we select the MFCCs as our feature vector to extract from. In addition, k-means clustering (Section 2.3.1) and Monte-Carlo sampling (Section 2.4.4) are also selected as the clustering and distance measure methods. To be more specific, three clusters are used in k-means algorithm and the number of sampling points in Monte-Carlo method is 3000.

For testing the functionality, each song is input as a query to the database and

thus calculate the song-by-song distance between pair. Because the final distance measure should fairly depend on the data length, we hereby normalize the largest distance to one, thus all the distances would be bounded in the interval of zero to one. Finally, we make the distance become symmetric by averaging

$$\text{dist}_{\text{symmetric}}(i,j) = (\text{dist}(i,j) + \text{dist}(j,i))/2 \tag{2.21}$$

for each i and j.

2.5.3 Results

Figure 2.6 shows a visualization of the 30×30 resulting distance matrix. Each element is displayed by a color in between the black and white. As illustrated by the color bar on the right side, the black represents zero distance while the white represents the maximal distance. The diagonal term of the matrix is totally black since the auto-term distance should be close to zero. Moreover, the elements around the diagonal line tend to be in the dark gray color (i.e., with shorter distance). This result may due to we arranged the songs with similar genre tags (e.g., Jazz and Bassa nova) or instruments to be close to each other.

To be more specific, we make an investigation on two particular entries in the distance matrix to verify the result. The first one is the 15th entry, *Waltz for Debby* by artist *Bill Evans*, which is shown in Figure 2.7. The height directly corresponds to the distance between the 15th song and the remaining songs inside the database. According to the plot, the three closest songs are listed as follows:

1. Jazz Trio – "Country" by Keith Jarrett
2. Jazz Vocal – "The Girl in the other Room" by Diana Krall
3. Bassanova – "C'est si bon" by Lisa Ona

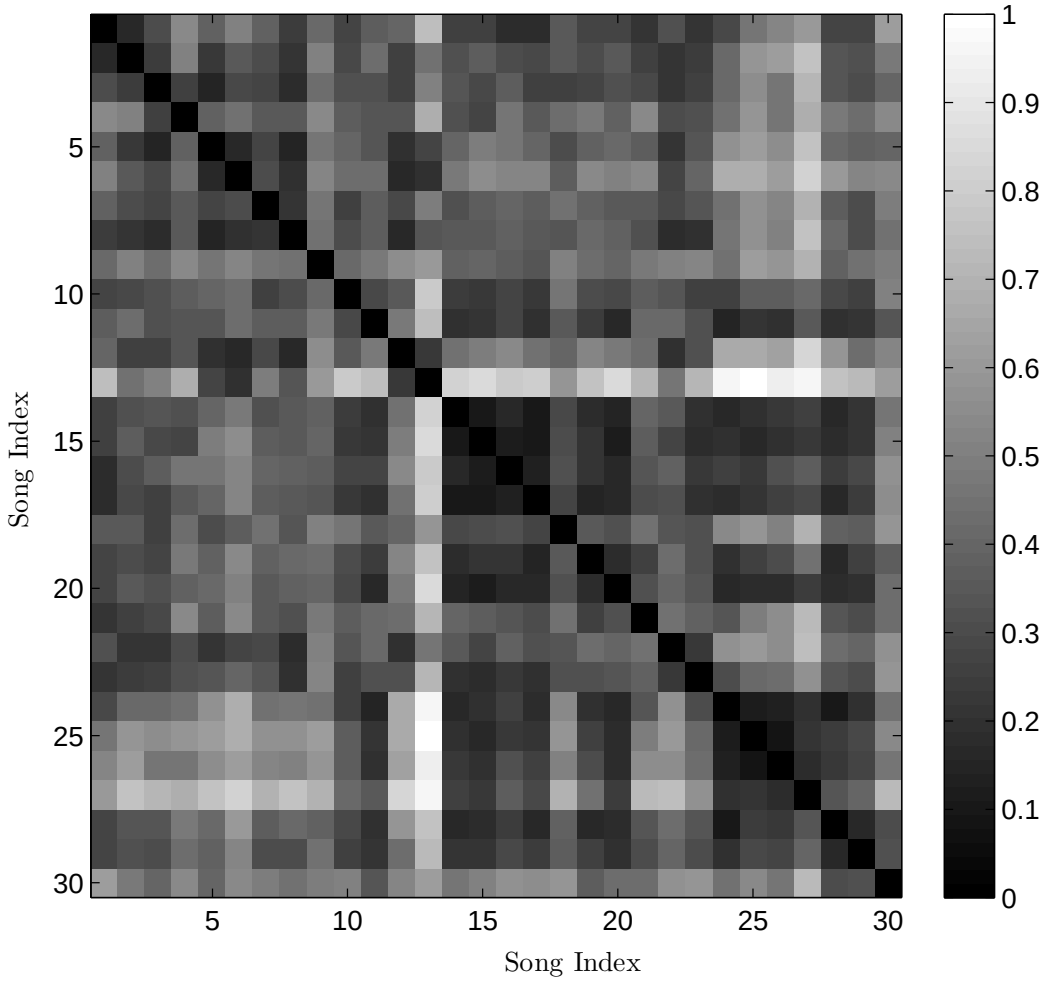


Figure 2.6: Distance matrix of the similarity measure experiment. Black represents zero distance while white represents the largest distance.

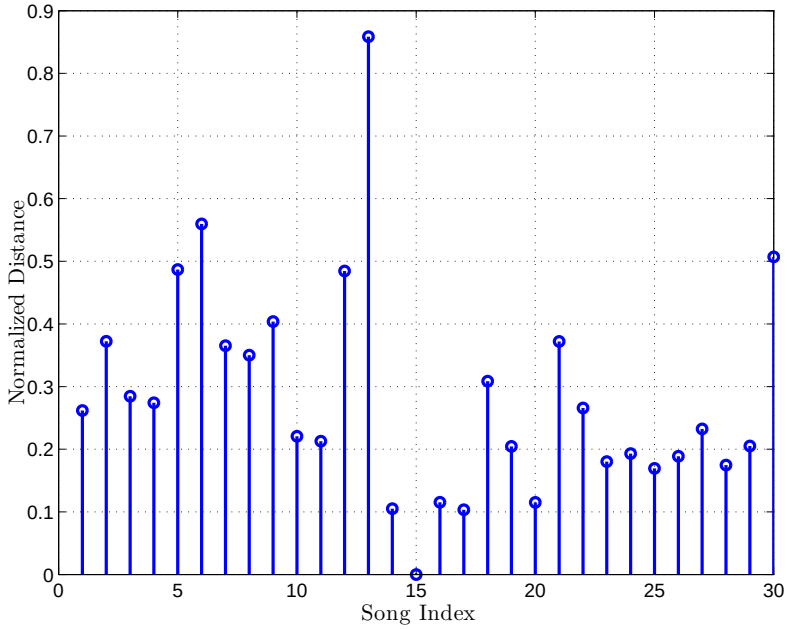


Figure 2.7: Distance vector of the 15th song, "Bill Evans: Waltz for Debby".

Based on the music theory, the genres of these songs are all considerably correlated to Jazz, but with different instrument arrangements. We also can find that the farthest one with this song is *Legacy of Kings* performed by *Hammerfall*. It is because the timbre and instrumentation of Jazz greatly differs from the metal style music.

For another, consider the 26th entry, *Piano Sonata No.3 mov.4* composed by *Chopin*, which is illustrated in Figure 2.8. The three closest songs related to it are as follows:

1. New Age Piano – "Pastoral" by Yuriko Nakamura
2. Japanese Ballad – "Crucify My Love" by X Japan
3. New Age Piano – "One Summer's Day" by Joe Hisaishi

Although these songs are labeled with different genres, it appears that they are all played on piano (i.e., consist the same instrument as the 26th entry). Note that the distance between it and the classical cello song (i.e., the 26th song) is somewhat

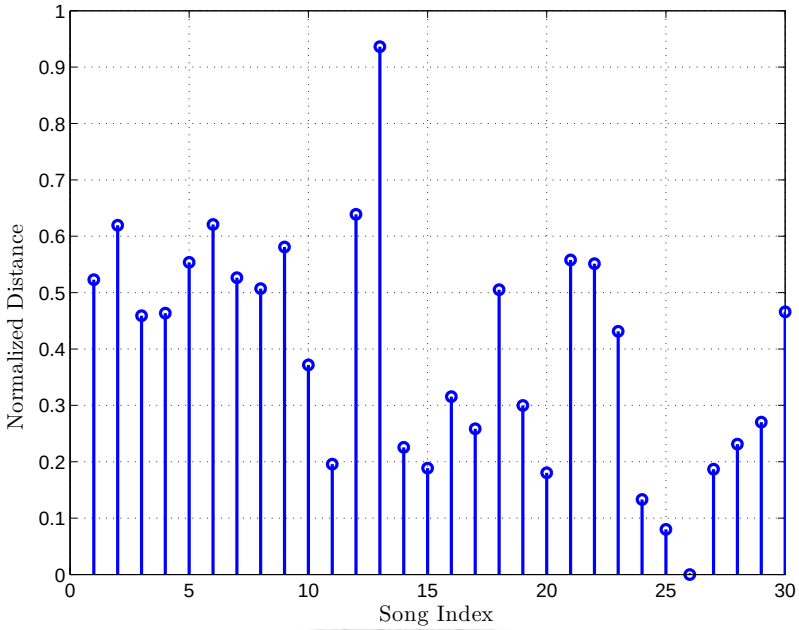


Figure 2.8: Distance vector of the 26th song, "Chopin: Piano Sonata No.3 mov.4".

larger than that with the new age piano music (i.e., the 24th song). The above two examples indicate that the similar songs calculated by the system tend to share the same genre or instrument arrangement with the query song, which is consistent with the human perception.

However, this two examples also reveal that the closest songs derived from the system does not aim at any of particular musical content. The relation between the retrieved songs may lie in the genre, instrumentation, or even the chord progression. The reason is that MFCCs cannot access any of these contents; instead, it only represents the spectral characteristic of each frame inside a song. Therefore, in the following chapter we will discuss how to derive a more selective feature that can be specifically directed against the instrumentation information.

We conclude this section by evaluating the Monte-Carlo sampling method in the cluster modeling step. As described previously, a large number of points are randomly sampled in the distributions, and then their likelihood function as a distance

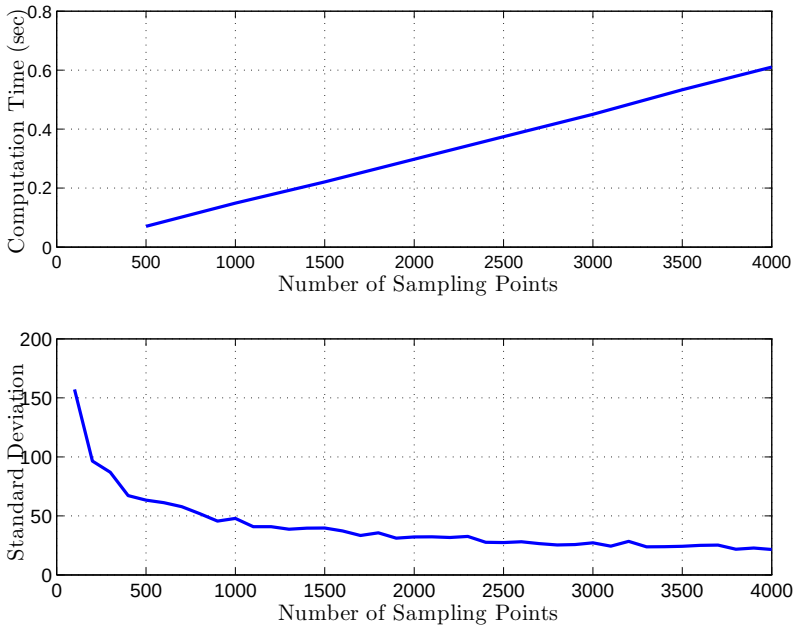


Figure 2.9: Evaluation of the Monte-Carlo sampling method. Top and bottom part represent the efficiency and performance analysis, respectively.

metric is calculated. In intuition we know that more sampling points should result in more precise distance estimation and much closer to the real likelihood in theory. But however it also leads to the enormous increase of the computation time. Following we aim at finding the relationship between the sampling points, the estimation precision, and the computation time. This can help us to find a reasonable number of sampling points in our experiment.

Here we choose two songs with their lengths approximately equal to three minutes for calculating the distance. They are both with the same format as previous: sampling rate 11025 Hz, mono wave file. There is only one variable in this experiment – the number of sampling points, denoted as NS. In order to calculate the precision of the estimation, the calculation according to each number will go for several times. The principle is, if it comes up with a lower precision, they may share a large standard deviation. Therefore we use the standard deviation of the trials to evaluate

the precision.

Figure 2.9 shows simulation result. The number of points is simulated from a very small number to 4000. According to the plot, the computation time increases linearly with number of points, while the standard deviation decreases. It can be shown that when the number of sampling points is approximately larger than 2500, it reaches a saturation point and does not go down anymore. So in this experiment we believe that $NS = 3000$ should yield an acceptable result with respect to saving the computation time.

2.6 Discussion

In this chapter, we implemented a music similarity measure system and used a music database that covers a variety of different styles to evaluate its performance. Although the result is considerably close to the human perception, the limitation is that it cannot be oriented toward any of the particular music content. Moreover, in order to utilize this measuring technique to a recommendation system, there are several notable points need to be considered.

The first one is the subjective nature of the similarity. People with different tastes would favor different styles of the music. Due to this fact, it is better to construct a personalized recommendation system with the maintaining of the preference file by the technique like relevance feedback [33]. The performance of the recommendation system can also be improved grouping the users with similar preference and interests together [5]. The second consideration is the great semantic gap between the high-level perception-related features (e.g., chord, key, and instrumentation) and the low-level features (e.g., MFCCs and spectral centroid). In Chapter 5, we will introduce

how to overcome the gap by designing a system to integrate the low-level features.

Table 2.3: List of the songs using in similarity measure experiments.

No.	Genre	Title	Artist
01	Rap/Hip-Hop	My Humps	The Black Eyed Peas
02	Rap/Hip-Hop	Don't Phunk My Heart	The Black Eyed Peas
03	Popular	All Rise	Blue
04	Rock	Fairy Tales Gone Bad	Sunrise Avenue
05	Rock	It's My Life	Bon Jovi
06	Rock	Earn the Crown	Backyard Babies
07	Rap/Metal/Rock	In the End	Linkin Park
08	Rap/Metal/Rock	Papercut	Linkin Park
09	PostRock	Rescue – Day 4	Explosions In The Sky
10	PostRock	Ett	Ef
11	Japanese Ballad	Crucify My Love	X Japan
12	Metal	Rusty Nails	X Japan
13	Metal	Legacy of Kings	HammerFall
14	Jazz Trio	Country	Keith Jarrett
15	Jazz Piano	Waltz for Debby	Bill Evans
16	Bossa Nova	C'est Si Bon	Lisa Ono
17	Jazz Vocal	Girl in the Other Room	Diana Krall
18	Fusion	Daisy Field	T-Square
19	Chinese Popular	A Little Love Piece	Sodagreen
20	Chinese Popular	Too Smart	Cheer Chen
21	Dance	Fast and Furious	Teriyaki Boyz
22	House	Da hype feat R.S.	DutchForce
23	Trance	Deadline	DutchForce
24	New Age	One Summer's Day	Joe Hisaishi
25	New Age	Pastoral	Yuriko Nakamura
26	Classical Piano	Piano Sonata No.3	Chopin
27	Classical Cello	Cello Suite No.1	Bach
28	Classical Symphony	The Planets	Holst
29	Classical Symphony	Symphony No.7	Beethoven
30	Electro Orchestra	Garlibdli Temple	Michiru Yamane

Chapter 3

Time-Frequency Analysis of Music Instrumental Signal

3.1 Introduction and Related Work

In digital signal processing (DSP), music signal is one of the most important categories that are appealing for research work. Many techniques in DSP have been applied to analyze the music signal in several applications. Consider the frequency analysis, the discrete Fourier transform (DFT) is a well-known and widely used method due to its good mathematical properties and a fast algorithm implementation: the fast Fourier transform (FFT). The complexity of calculating a N-point FFT is only in the order of $N \log_2 N$.

However, music signal has its own unique characteristics that are considerably dissimilar to other signals. For instance, comparing to the speech signal, generally the music signal is much complex for processing in many aspects. First of all, in music signal lots of instruments are played simultaneously, while in speech signal

only one speaker is considered at the same time. Secondly, instruments can be classified according to how the sound is initially produced (e.g., string, percussion and wind families). Each of these instruments preserves different mechanism, while in speech signal the generating model is fixed (e.g., the vocal tract model).

Moreover, an isolated note of a single musical instrumental tone usually comprises a fundamental frequency component and successive harmonics as a pulse train in frequency domain. As a result, Brown [34] introduced the constant Q transform aiming to fit these special attributes of the musical signal. The calculation is with a constant ratio of center frequency to resolution, since that the frequencies have been chosen to integrate the scale of Western music are geometrically distributed. To compensate its long computation time, Brown improved the time-consuming calculation by incorporating the FFT algorithm within the matrix multiplication step [35]. Nevertheless, the calculation efficiency is still an essential issue. To address the channel selectivity and the complexity problems, there are several derivations which combine the constant Q transform with other filter structures such like the fast filter bank (FFB) and the bounded-Q transform [36], [37], [38]. In addition, these transforms have been applied to some music-related applications by researchers [39], [40].

Another drawback of the conventional DFT is that it cannot observe the spectrum varying according to time. It is still based on calculating the mapping from time to frequency domain, instead of consider the whole time-frequency plane. Music signal, due to its highly non-stationary property, requires a good resolution both in the time and frequency domain. To overcome this problem, one simple solution is to segment the signal into frames and then calculate the short-time Fourier transform (STFT). The techniques in time-frequency analysis such like the Gabor

transform and the wavelet transform are also suitable for analyzing the signal. In [41] Pielemieier introduced some of the properties and applications when applying time-frequency analysis to the musical signal.

The aim of this chapter is to combine the idea from the constant Q transform and the time-frequency techniques to analyze the musical instrumental signal. The rest of this chapter is organized as follows. At the beginning, a brief introduction of musical instrumental signal characteristics is presented. In this section we stress on the related characteristics such like the pitch and the harmonics of the musical signal. Secondly, the idea of the constant Q transform is reviewed and a comparison with the traditional DFT is held. After that, the transform is extended into the time-frequency distribution function and a simple implementation method is presented. Finally, we apply this transform to three major instrumental signals to verify the performance.

3.2 Characteristics of Musical Instrumental Signal

The musical signal is a special class in the signal category that has its own characteristics different from the speech signal in many ways. First of all, music normally has a wide range frequency distribution among the audible range of human, from 0 to 20kHz. As we know the bandwidth of the speech signal is usually limited into 50 to 7kHz. In addition, when considering time-domain characteristics, musical signal usually has a lower silence ratio except that it is sung by a singer or played on a solo instrument only. Comparing to an ordinary speech signal, music has lower

Table 3.1: Illustration of the pitch and music notation.

Note	C4	#C4	D4	#D4	E4	F4
Frequency (Hz)	261.6	277.2	293.7	311.1	329.6	349.2
Note	#F4	G4	#G4	A4	#A4	B4
Frequency (Hz)	370.0	392.0	415.3	440.0	466.16	493.9

variability in zero-crossing rate (ZCR) which is defined as

$$\text{ZCR} = \frac{1}{2N}\sum_{n=1}^{N}|\text{sign } x(n) - \text{sign } x(n-1)| \tag{3.1}$$

Following we introduce two main characteristics: the pitch and the harmonics of the music signal.

3.2.1 Pitch

In Western music theory, from the age of musician Bach (1685-1750), the music sound we hear is grouped into twelve separate categories. Each of the category is labeled by a unique representation called *note*. That is to say, the letters A through G represent seven notes and the other five semi-tones are represented by appending either a pound sign (#, or sharp) or something that looks remarkably similar to a lower-case b (also called a flat).

Each tone has its own pitch that is related to the fundamental frequency (FF). Musicians found that a double fundamental frequency sounds the same as its original note in human perception, hence they defined an *octave*. For instance, C4 (also the middle C) means the 4th octave of the C note in each instrument. The corresponding fundamental frequency of each semi-tone is defined in a geometrically distributed way as

$$f_{\text{n}^{\text{th}}\text{ note}} = f_0 \times 2^{\frac{n}{12}} \tag{3.2}$$

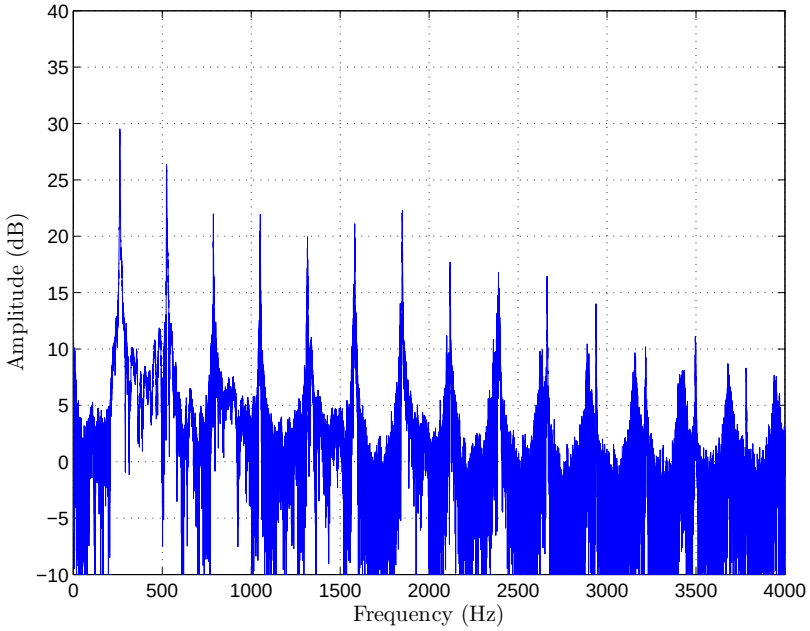


Figure 3.1: Spectrum of a C4 piano note.

where f_0 is the base frequency. The fundamental frequency components of semi-tones are not equally-spaced in frequency domain. However, they distribute like a log-scale instead of the linear distribution. Table 3.1 lists an illustration of the pitches and their corresponding frequencies within an octave.

3.2.2 Harmonics

So far we know the corresponding fundamental frequency of each semi-tone, but it is still not enough to describe the characteristics of a note for a specific instrument. Any non-electronic instrument actually produces many frequencies, all of which are overshadowed by the fundamental frequency [42]. These extra frequencies are called the *harmonics*. For example, Figure 3.1 shows the spectrum of a C4 note played on piano. It can easily observed that the first peak of the amplitude is centered at 261.6Hz as its fundamental frequency, but with a set of additional peaks follows on.

The successive harmonics component is also called *partial* in music theory. In

most cases they are an integer multiple of the fundamental frequency. Although in this example the power of partials tends to decay when frequency increases, it is not a general phenomenon. On the contrary, it has been discovered that different instrumental tones consists of different partial distributions. One can use this distribution difference to discriminate the desired instrument from others. This technique has been utilized in the unsupervised music source separation applications [43].

3.3 Constant Q Transform

3.3.1 Motivation

The conventional spectral transformation such as the DFT takes N-tuple points into account, and then transform the signal to frequency domain for further analysis. It yields several well-known properties, such like the equally-spaced frequency resolution, and the linear scale in frequency domain. But as described before, the musical signal comprises several pulses in frequency domain: the fundamental frequency and the successive partials. In Figure 3.1 we can find that after applying the DFT, several peaks are presented and equal-spaced distributed. For the partial tracking problem, it would be better to do the transform against the log scale to obtain a constant pattern in the frequency domain for regular pattern recognition. In addition, if we utilize the mapping from the DFT into a log-scale frequency axis, it gives too little information about the low frequency part and too detailed information at the high frequencies.

The second consideration is about the frequency resolution. The frequencies that have been chosen to make up the scale of the Western music are geometrically

Table 3.2: Comparison of variables in calculation of the DFT and the CQT.

	CQT		**DFT**	
Frequency f_k	$f_0 \cdot 2^{\frac{k}{b}}$	exponential in k	$k \cdot \Delta f$	linear in k
Window Size N_k	$R \cdot Q/f_k$	variable	N	constant
Resolution Δf_k	f_k/Q	variable	R/N	constant
$f_k/\Delta f_k$	Q	constant	k	variable

spaced. In this case the frequencies are located in the specific bin once the note (or pitch) fixed. However, the DFT does not invoke such information to transform the musical signal. Instead of that, the resolution should be geometrically related to the frequency. For instance, consider two adjacent notes in different octaves. The first pair is A4 with 440 Hz and #A4 with 466.16 Hz, thus the frequency difference between them is 26.2 Hz. On the other hand, if we consider two octaves below: A2 with 110 Hz and #A2 with 116.54 Hz, the frequency difference 6.54 Hz is much smaller than the former one. From this example we can figure out that due to the log-scale distribution of the fundamental frequencies of musical notes, the higher frequency resolution is needed for the higher frequency component. This property also resembles the situation in our auditory system.

The constant Q transform (CQT) is proposed to solve above issues by Brown in [34]. Table 3.2 summarizes a comparison of different variables in calculation of the DFT and the CQT.

3.3.2 Implementation

Like the DFT, a CQT also consists of a bank of filters. But in contrast with the former, the filterbank has geometrically spaced center frequencies f_k,

$$f_k = f_0 \cdot 2^{\frac{k}{b}} \tag{3.3}$$

where b stands for the number of filters per octave and f_0 is the minimal center frequency. To fulfill the previous requirement, here we choose the bandwidth of the k-th filter as

$$\Delta_k = f_{k+1} - f_k = f_k(2^{\frac{1}{b}} - 1) \tag{3.4}$$

This yields a constant ratio of bin frequency to resolution

$$Q = \frac{f_k}{\Delta_k} = (2^{\frac{1}{b}} - 1)^{-1} \tag{3.5}$$

Because the bandwidth of the filter also equals to

$$\Delta = \frac{f_s}{N} \tag{3.6}$$

where f_s is the sampling frequency and N is the number of points in window. Thus the desired bandwidth (3.4) can be achieved by choosing the window length to be

$$N_k = Q\frac{f_s}{f_k} \tag{3.7}$$

For other parameters, first we need to choose a minimal frequency f_0 and the number of bins per octave b according to the requirement of the application. To illustrate, $b = 24$ yields a quarter-tone resolution transform. In addition, the maximal frequency $f_{\max}$ only affects the number of bins to be calculated. The CQT can thus be implemented as following:

$$X^{cq}[k] = \frac{1}{N_k}\sum_{n=0}^{N_k} x[n]w_{N_k}[n]e^{\frac{-2\pi jnQ}{N_k}} \tag{3.8}$$

where

$$K = b \cdot \log_2(\frac{f_{\max}}{f_0})$$

$$Q = (2^{\frac{1}{b}} - 1)^{-1}$$

and for $k < K$

$$N_k = Q\frac{f_s}{f_k}$$

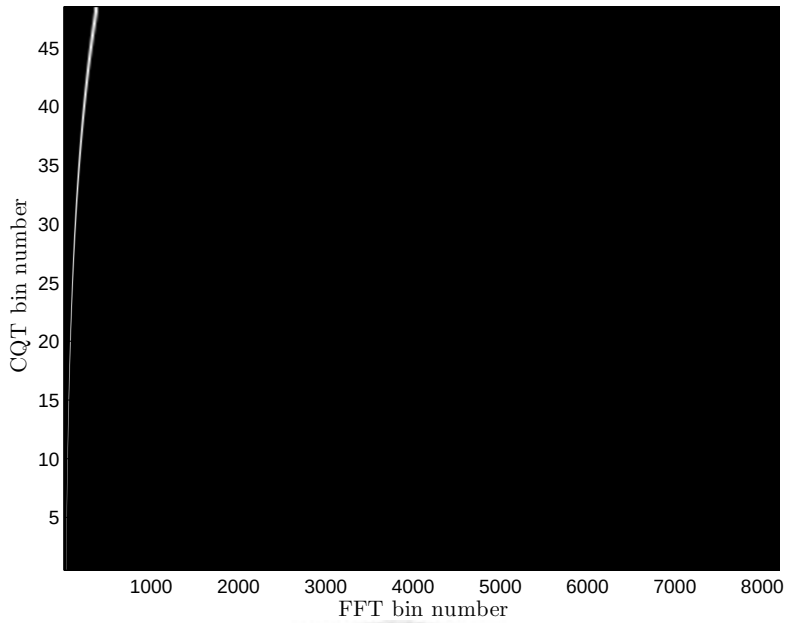


Figure 3.2: Magnitude of the spectral kernel in the CQT. The CQT here is with sampling frequency 44100 Hz and quarter-tone resolution.

For the window function $w_{N_k}[n]$, Brown used the Hamming window instead of the rectangular window in his implementation.

We should also notice that the CQT as in (3.8) is not invertible. Since the temporal decimation factor $N[k]$ is greater than the analysis window length $w_N[n]$ for high-frequency component. This means that there are some samples never being used when calculating the higher frequency bins.

3.3.3 An Efficient Algorithm

Comparing to the conventional FFT implementation, although it possesses several properties that are pretty suitable for musical signal analysis, the calculation of the CQT according to (3.8) is considerably time consuming, thus an efficient algorithm is highly needed.

We can comprehend the CQT as a matrix multiplication form

$$X^{cq} = x \cdot T^* \tag{3.9}$$

where T^* is the complex conjugate of the temporal kernel

$$T_{nk} = \begin{cases} \frac{1}{N_k} w_{N_k}[n] e^{\frac{-2\pi j n Q}{N_k}}, & \text{if } n < N_k \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

Since the temporal kernel T is independent of the content of signal x, it can thus be pre-calculated to speed up the total computation. Nevertheless, it is pretty memory-consuming to store the whole matrix value. Because that T contains many non-vanishing value, computing the result of $x \cdot T^*$ still takes time.

Brown invented a reduction method in [35] to further reduce the computation time. It can be shown that for any two discrete functions $x[n]$ and $y[n]$,

$$\sum_{n=0}^{N-1} x[n] y^*[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] Y^*[k] \tag{3.11}$$

where $X[k]$ and $Y[k]$ denote the DFT of $x[n]$ and $y[n]$, respectively. This is a form of the Parseval's equation. The idea is to carry out the matrix multiplication in the frequency domain. Since the windowed complex exponentials of the temporal kernel have a DFT that vanishes almost everywhere except for the immediate vicinity of the corresponding frequency. The spectral kernel is a *sparse matrix* as depicted in Figure 3.2. Brown suggests to eliminate components that are below the threshold with accepting a considerably small error.

As a result we can calculate the CQT in a different way:

$$X^{cq}[k] = \frac{1}{N} \sum_{n=0}^{N-1} x^{ft}[n] S_{nk}^* \tag{3.12}$$

where S_{nk} consists of one dimensional DFTs applied column-wise. This equation involves with less computation than the former one. Although it takes time to

create the spectral kernel, but once it has been done then all the succeeding CQT computations can be performed much faster.

3.4 Time-Frequency Analysis Using the Constant Q Transform

In previous section we have introduced the procedure of calculating the CQT. However, if we want to apply it to a real musical signal, say a three to four minutes wave file, there are some problems still needs to be solved. The first one is, as the same consideration as the DFT, we are not able to discover the frequency distribution varying with respect to time. To illustrate, for a piano record consisting of a series of melody, the result of the CQT should be fairly complicated, and it makes no sense to analyze the individual notes produced by the instrument. This can be easily solved by utilizing the concept from the short-time Fourier transform (STFT) or Gabor transform. The idea is to divide the signal into several pieces in time, and to do the spectrum transform for each segment separately. Nevertheless, the CQT somewhat behaves differently as that DFT does. In CQT, the determination of $f_{\min}$ and $f_{\max}$ make the number of points inside a window become fixed. For example, consider the situation that sampling rate equals to 32000 samples/s, $f_{\min} = 175$ Hz yields a window with 6231 samples, while $f_{\max} = 13432$ Hz yields a window with only 162 samples. Thus for implementing the time-frequency version of the CQT (denoted as TF-CQT), it is reasonable to determine the sliding window shift size B according to $f_{\min}$ and $f_{\max}$.

The second problem is, if we choose the sliding window shift size to be equal

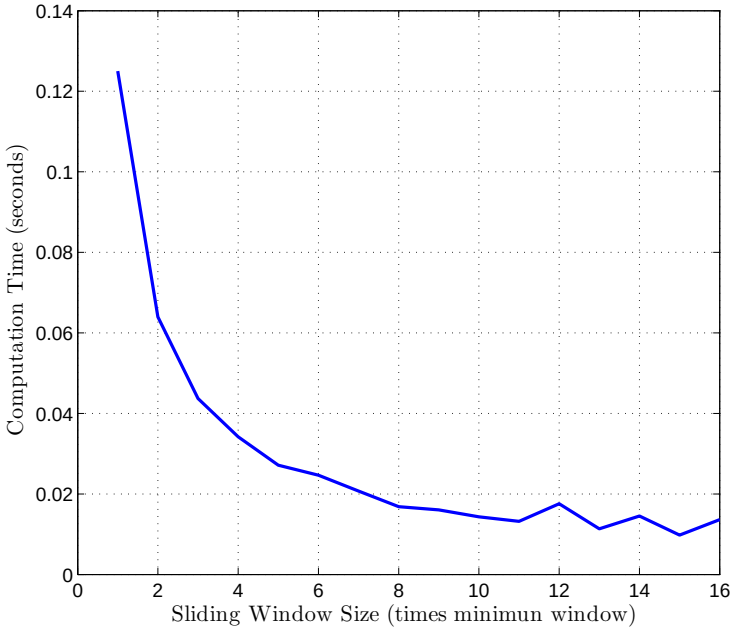


Figure 3.3: Computation time of the TF-CQT in terms of different k. The input signal is a 2.2-minute wave file, with $f_s = 44100$ Hz.

to the number of sampling points corresponding to $f_{\min}$, we would lose a lot of information in the higher frequency component. To address this problem, we define the minimum required shift size $B_{\min}$ the same as the window size of $f_{\max}$, that is

$$B_{\min} = \frac{f_s \cdot Q}{f_{\max}} \tag{3.13}$$

In this implementation we set the sliding window shift size B to be a multiple of $B_{\min}$

$$B = k \cdot B_{\min} \tag{3.14}$$

where k is an integer multiplier. Thus the CQT in (3.8) is extended into the TF-CQT by

$$X^{\mathrm{TF-CQT}}[n,k] = \frac{1}{N_k} \sum_{p=nB}^{nB+N_k} x[p] w_{N_k}[p] e^{\frac{-2\pi j p Q}{N_k}} \tag{3.15}$$

The choosing of multiplier k in (3.15) is a trade-off problem between the simulation time and the time resolution. The lower value of k would leads to longer

computation time but with more overlapped frequency components and detailed time-varying information. Figure 3.3 shows the required computation time of the TF-CQT with varying values of k. The computation time tends to decrease exponentially while the sliding window size increases.

3.5 Simulation Results

In this section the TF-CQT is applied to analyze the real-world musical instrumental signal. In our implementation, b is set to be equal to 12, which results in a semi-tone resolution per octave. The minimal frequency and maximum frequency are chosen to be 130.8 Hz (C3 note) and 2092.8 Hz (C7 note), respectively.

3.5.1 Music Database

The musical instrumental samples are collected from the *Electronic Music Studios*[1] provided by the University of Iowa. It offers a wide range of isolated notes played on several different instruments, such like piano, violin, oboe, bassoon and so on. The detail of the database that we construct can be referred to Table 4.1. The resource is allowed to be downloaded and used in research purpose for free. Originally, all files are stored in 16-bit, 44.1 kHz, mono, AIFF format. We then converted them to a 8-bit, 44.1 kHz, wav format in our experiment.

In this simulation we select piano, viola and flute to construct the instrument set, since they all have different sound producing mechanisms and are classified to different instrument families (will be discussed in Chapter 4). The instruments are played on two notes: C4 (261.6 Hz) and E4 (329.6 Hz). All the wave files from single

[1] http://theremin.music.uiowa.edu/

notes are trimmed into 2.2 seconds in order to unify the duration.

3.5.2 Results

Figure 3.4, 3.6 and 3.8 show the results of applying the TF-CQT to three instruments played on the C4 note. To evaluate the result, we also plot the spectrogram generating by the STFT in Figure 3.5, 3.7 and 3.9. In STFT, a 4096-points Hamming window with half overlapping is employed. Since the sampling frequency is considerably high, we only plot the first 120 FFT-bins to observe the spectrum peaks. In comparison with the STFT results, the TF-CQT requires less bin number, and each bin is corresponding to a specific semi-tone. This makes TF-CQT suitable to deal with the musical instrumental signal. For another, the spectrum peaks in the STFT result are equally-distributed while the peaks in the TF-CQT are logarithmic distributed. In the partial-tracking problem, it is more suitable to use the TF-CQT instead of the STFT. The reason is, although the tone differs, the pattern in the TF-CQT remains the same with only a shift in the frequency domain. This logarithmic distribution makes the template to be invariant to the tone change.

According to the plots, it is easy to find out the difference between these instrumental signals through their time-frequency distribution. Consider the time axis, the piano, which can be regarded as a percussion instrument, have its energy decaying rapidly after the hammer hits the string inside. The time-frequency distribution in the other two signals can somewhat sustain for a while until the sound-producing process is finished. Next, consider the frequency axis, we discover that each instrumental signal possesses its own energy distribution in the harmonics. Besides, in some instruments the harmonic energy is even greater than the fundamental fre-

quency energy.

To be more specific, in Figure 3.10 we draw the energy curves of three major CQ-bins: the FF, 1st and 2nd partial, but this time with two different observation notes: C4 and E4. The result shows that even though the tone varies, the shapes from same instrument still resemble closely to each other. This makes the TF-CQT to be a powerful tool for instrument classification applications.

3.6 Discussion

In this chapter, we extend the idea of the CQT to the time-frequency plane and use it to analyze several instrumental signals. Due to its high compatibility with the music signal, the result shows that it is more reasonable to use the TF-CQT in the audio applications. However, there are still some disadvantages and limitations that needs to be discussed in the end of this chapter. First of all, although the CQT is suitable for dealing with the pattern recognition problems, the CQ-bin cannot exactly match the frequencies of the successive partials. For example, the fundamental frequency of the A4 note $f_{A4,1}$ is 440 Hz, and the frequencies of the second $f_{A4,2}$ and the third $f_{A4,3}$ harmonics are 880 Hz and 1320 Hz, respectively. $f_{A4,2}$ is exactly equal to the fundamental frequency of the A5 note, thus resulting in a correct match to the CQ-bin. Nevertheless, there is no CQ-bin with its central frequency being exactly the same as $f_{A4,3}$. The closest two bins are 1318.5 Hz and 1396.9 Hz, if we use semi-tone resolution. For the second consideration, the computational efficiency for the CQT is still an issue that needs to be discussed and improved by other techniques. Comparing to the STFT, it still takes too large computation time.

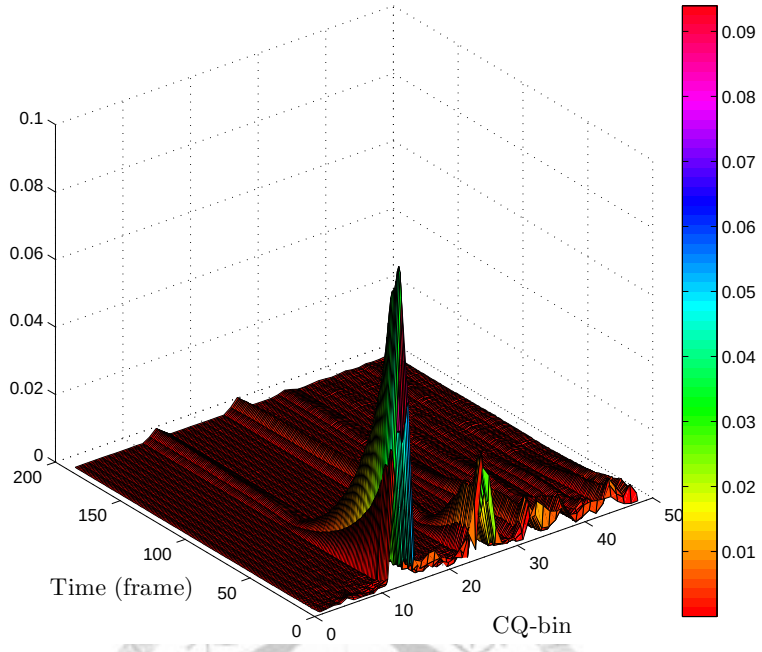


Figure 3.4: TF-CQT result of the C4 note played on piano. $B = B_{\min}$.

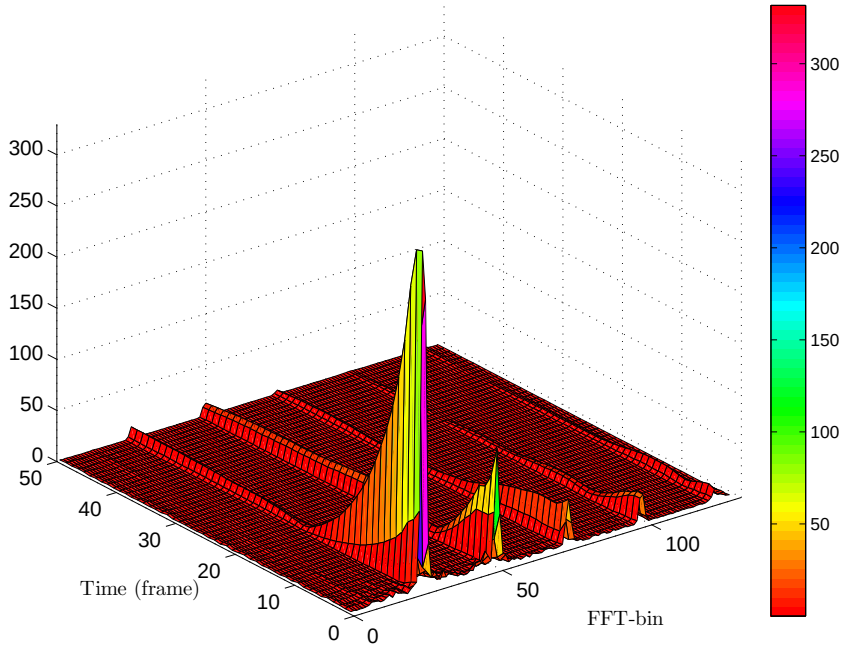


Figure 3.5: STFT result of the C4 note played on piano. A 4096-points Hamming window with half overlapping is used.

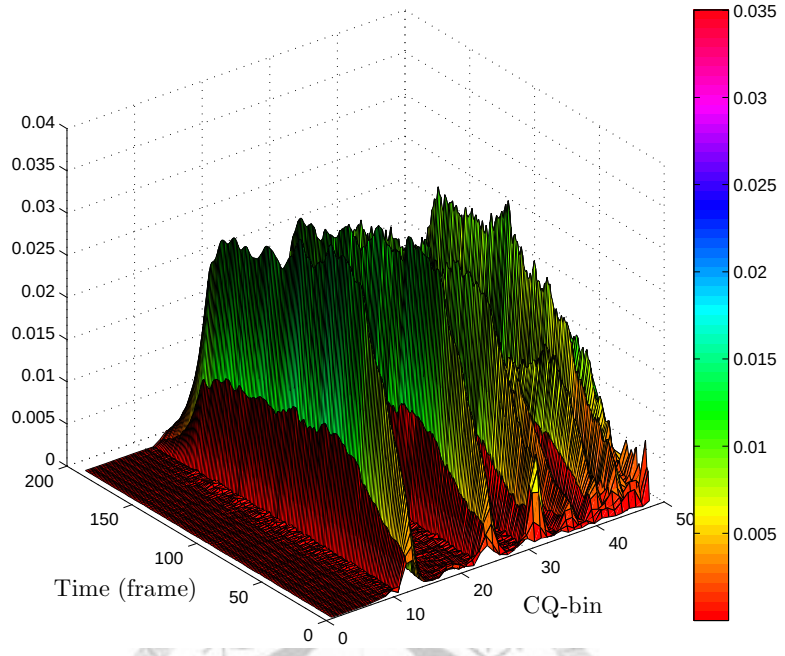


Figure 3.6: TF-CQT result of the C4 note played on viola. $B = B_{\min}$.

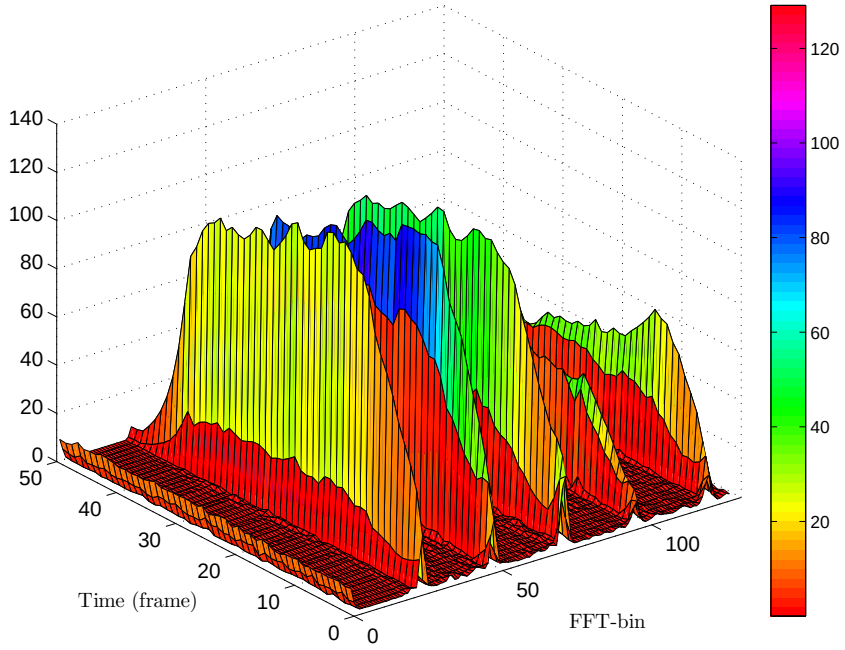


Figure 3.7: STFT result of the C4 note played on flute. A 4096-points Hamming window with half overlapping is used.

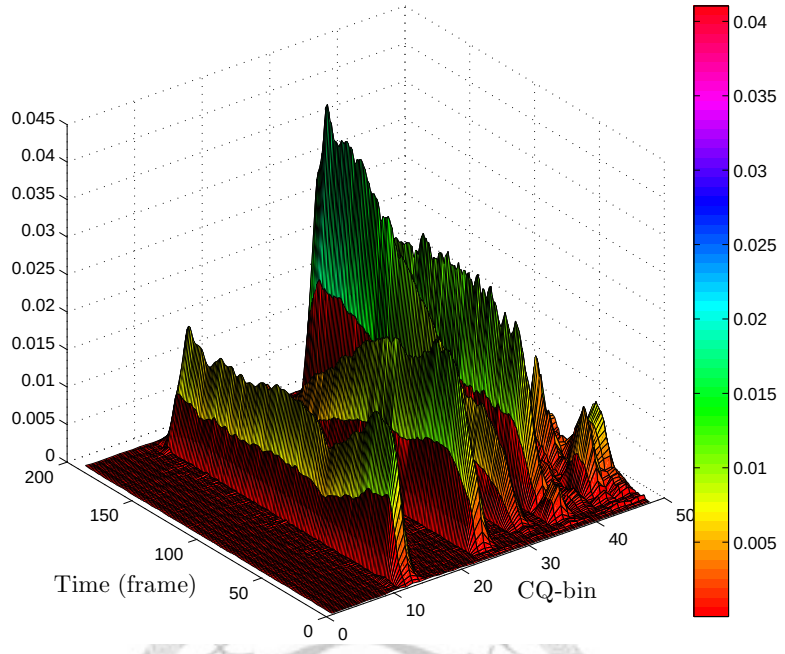


Figure 3.8: TF-CQT result of the C4 note played on flute. $B = B_{\min}$.

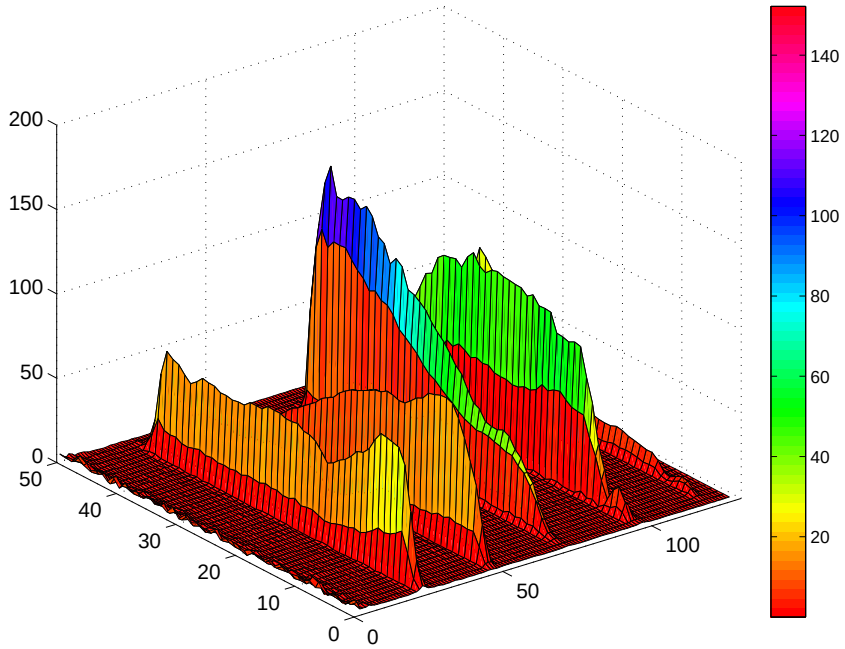


Figure 3.9: STFT result of the C4 note played on flute. A 4096-points Hamming window with half overlapping is used.

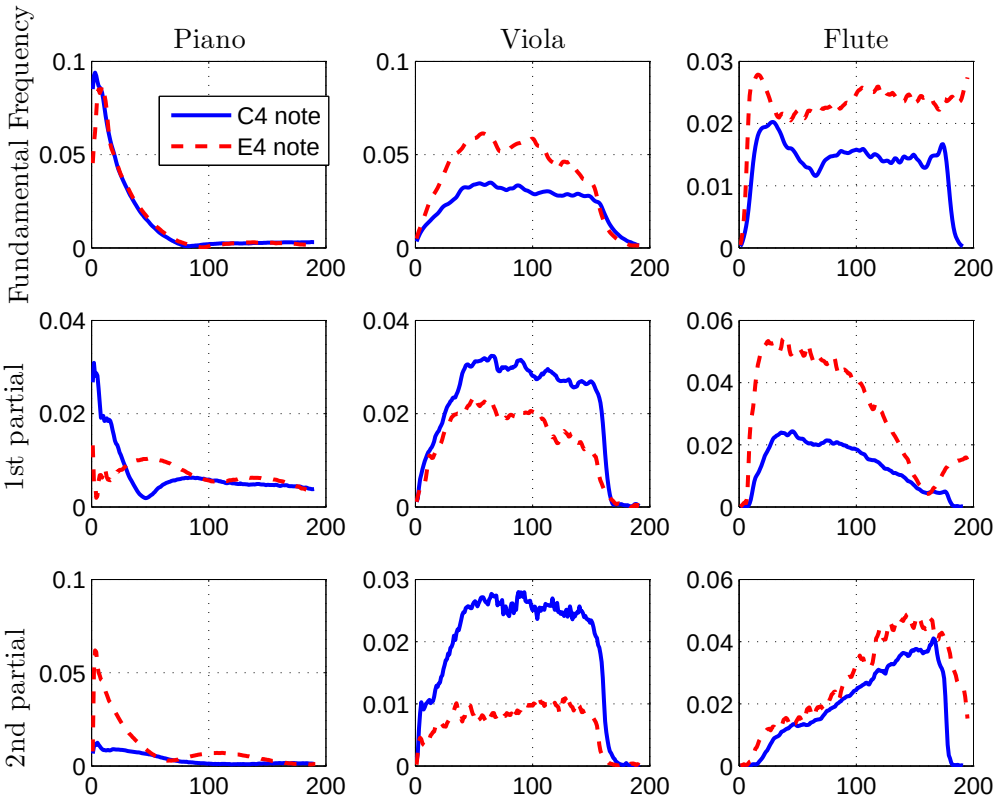


Figure 3.10: TF-CQT result of the C4 and E4 note played on three instruments, only three major bins (FF, 1st and 2nd partial) are considered. The x-axis represents the frame, while y-axis represents magnitude.

Chapter 4

Instrument Classification of Monophonic Music

4.1 Introduction and Related Work

Based on the findings in the previous chapter, it seems that the harmonic structures are able to represent the various instrumental signals and can be further utilized to make automatic classifications. During the past few decades, the automatic classification of instrumental signals has become a fascinating and essential subproblem in content-based music information retrieval tasks. It is closely related to the area of machine learning and signal processing. However, the musical instrument identification has not yet received much interest from researchers comparing to the speech recognition, for instance. The goal is to correctly classifying the instrumental signals according to their timbre variations.

Indeed, there are several studies conducted by researchers toward this topic. Eronen *et al.* reported a comprehensive study on classifying the orchestral instruments

using the spectral and temporal properties of sounds [44]. There database covers 1498 samples and is tested in terms of both family and individual classifications. Various types of features, such like ceptral and linear prediction coefficients, are compared and ranked by the output recognition rates of the classifiers [45]. Krishna *et al.* applied the line spectral frequencies (LSF) related features, which are proved to be effective in speech applications, into the classification work toward the solo phrases of the instruments [46]. Similarly, Essid *et al.* adopted the conventional statistical approaches, Gaussian Mixture Models (GMMs) and support vector machines (SVMs) to deal with the classification works of the solo excerpts [47]. Since these conventional feature extraction methods and classifiers have gradually been well investigated, researchers began to design new feature sets and classification schemes. For example, Benetos *et al.* developed a subset feature selection methods based on the non-negative matrix factorization (NMF) [48]. Essid *et al.* applied the pairwise classification strategies to improve the classification perofrmance [49]. Moreover, Klapuri analyzed the instrumental signals of the isolated notes using the source-filter-decay model [50]. The classification is done by estimating the coefficients of the testing instrumental signals. Su implemented a musical auto-transcription system, which consists of several modules like the rhythm recognition unit, pitch recognition unit and timbre recognition unit [51]. The system aims at automatically transcript the musical content inside a music clip by integrating a set of modules together.

The aim of the study in this chapter is to illustrate the taxonomy of the instrument hierarchy in literature, and then conduct an experiment that mainly deals with the instrumental signals of isolated notes. The rest of this chapter is organized as follows. In Section 4.2 we will make a literature review of the history of musical

instrument classification, and specify the taxonomy adopted in this work. Next, the detail of the automatic classification system is introduced in Section 4.3. Finally, the experimental results and the discussion are drawn in the end of this chapter.

4.2 History and Concept of Musical Instrument Classification

At various times, in various different regions, various schemes of musical instrument classification have been proposed. Kartomi introduced the nature, the society-oriented classification toward literary and oral transmission in [52]. In his work, the instrument classification methods applied in several regions (e.g., Chinese, Indian, Srilankan, and Java) have been comprehensively investigated. The oldest known classification scheme can be traced back to Chinese in fourth century BC. In that approach, the instruments are grouped together according to what they are made of, e.g., the stone, wood, and silk.

An ancient system has been proposed in first century BC originally for Indian instruments. The instruments are divided into four groups, mainly according to how the sound is originally produced and the materials: whether the sound is produced by vibrating string, by vibrating columns of air, percussion instrument made of wood or metal, and percussion instrument made with skin head. The method was revised by Hornbostel and Sachs in 1914. The system, named as *Hornbostel-Sachs system*, partition the instrument into four categories as follows.

1. *Idiophones*, the instruments which the sound is produced by vibrating themselves, such like the xylophone.

2. *Membranophones*, the instruments which the sound is produced by vibrating membrane, such like the drums.

3. *Chordophones*, the instruments which the sound is produced by vibrating strings, such like the piano and violin.

4. *Aerophones*, the instruments which the sound is produced by vibrating the columns of air, such like the organ and the oboe.

It is also possible to classify the instruments by the pitch range. The following terms are named after the singing voice classification.

1. *Soprano* instruments, such like the flute, clarinet, violin and trumpet.

2. *Alto* instruments, such like the oboe, alto flute, viola and horn.

3. *Tenor* instruments, such like the trombone.

4. *Bass* instruments, such like the bassoon, double bass, bass clarinet and tuba.

Generally, the most widely used system in the West today, should be the one that divides instrument into the wind, strings and percussion families. In some cases the wind family is further divided into the woodwind (i.e., wind instruments with a reed) and the brass instrument (i.e., wind instrument where the air is set in motion directly by the lips). Figure 4.1 shows the taxonomy of the instrument classification adopted in this work. As can be seen, it represents as a double-layer taxonomy, which combines the instrument family (e.g., the woodwind) as the first layer, and the individual instrument (e.g., the flute) as the second layer. In the aspect of the machine learning problem, the recognition rates can be separately defined in these two layers, one for instrument family and another for individual instruments.

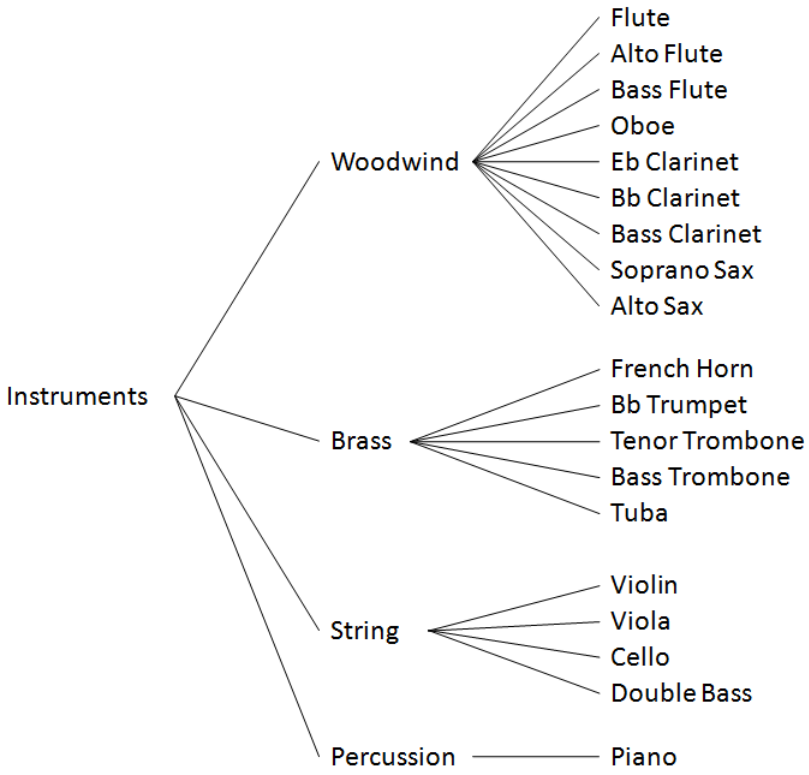


Figure 4.1: Taxonomy of the instrument classification opted in this work.

The double-layer nature of the taxonomy also improves the classification capability comparing to the approach that directly classifies the individual instruments.

4.3 Description of the Proposed System

Figure 4.2 shows the block diagram of the instrument recognition system. The system comprises the conventional machine learning functionalities: the training data with correct labels are first applied to establish models which can be used to determine the estimated labels of the testing data. Basically, the system is similar to the one proposed by Deng *et al.* [53], except for the feature selection and ranking processes.

In the feature extraction step, the MFCCs and the MPEG-7 Audio Timbre De-

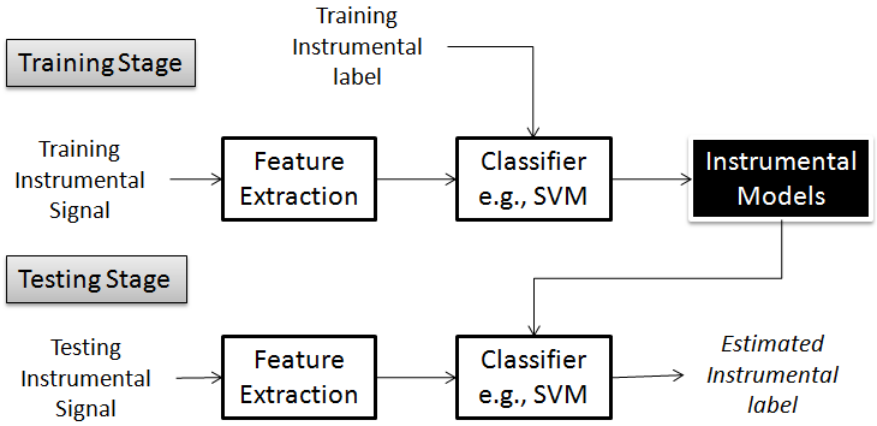


Figure 4.2: Block diagram of the instrument classification system.

scriptors are extracted from every instrumental signal and form a 33-dimensional low-level feature vector, which is the same as the one we used in Chapter 5. The classifiers chosen in this experiment is the support vector machines (will be illustrated in Section 4.3.2). The experimental data is first divided into the training and testing subset in order to evaluate the recognition performance. The k-fold cross validation method (will be introduced in Section 4.3.3) is used in this simulation.

4.3.1 Feature Normalization

In order to enhance the performance of classifying the signals, a feature normalization process is required prior to the SVM classifiers [54]. There are two well-known normalization methods can be applied in this stage. The first one is to normalize the feature vector into a zero-mean and unit standard deviation vectors. Suppose we have a set of training data $x_1, x_2, ..., x_N$. Let $\mathbf{v}_i$ denote the feature vector extracted from the ith data x_i, which is a p-dimensional vector. First we need to calculate the mean and variance of every feature vector in a particular dimension j, denoted as

μ_j and σ_j, respectively. Then the normalized value of $\mathbf{v}_{i,j}$ is calculated by

$$\mathbf{v}^*_{i,j} = \frac{\mathbf{v}_{i,j} - \mu_j}{\sigma_j} \tag{4.1}$$

The normalization process is repeated for every particular dimension in the vector space.

The second method is linear normalization, which normalizes the feature vector into a distribution in between of zero and one. That is,

$$\mathbf{v}^*_{i,j} = \frac{\mathbf{v}_{i,j} - \min_j}{\max_j - \min_j} \tag{4.2}$$

where $\max_j$ and $\min_j$ denotes the maximal and minimal value appearing in the jth dimension of the feature vectors. In this study, we adopt the first one as our feature normalization method.

4.3.2 Support Vector Machine

For a series of data, in some applications we would like to partition them into two sets, according to their features. Various approaches have been proposed by researchers, like the k-nearest neighbor, neural network and decision tree, all result in similar recognition rates. However, the support vector machine possesses simple nature and is easy to use in most of the applications. The support vector machine (SVM) is a supervised learning algorithm designed for classification and regression problems. The input data points are considered as two sets of vectors in an N-dimensional space. The idea of an SVM is to construct a separating hyperplane in such space, denoted as the *optimal separating hyperplane*, which maximizes the margin between the two corresponding data sets. In order to calculate the margin, two parallel hyperplanes, denoted as the *support hyperplane* are constructed, one on

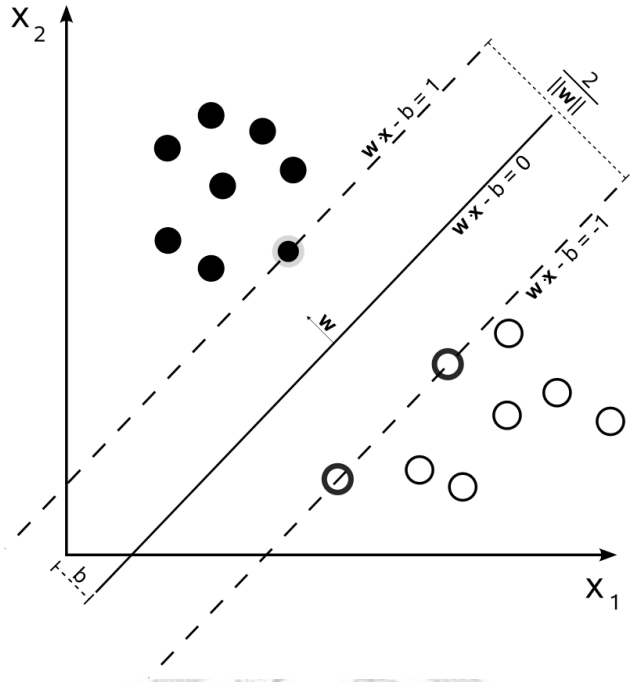


Figure 4.3: Illustration of the SVM. Black and white points are corresponding to different data set.

each side of the separating hyperplane, which are pushed up against the two data sets. In intuition, a good separation is achieved by the hyperplane that has the largest distance to the nearest data points of both sets, since generally the larger the margin will give less recognition error.

Figure 4.3 gives an illustration of the SVM classification. In the 2-D space, black and white points correspond to different data sets. The goal is to find a line with the largest margin to discriminate the training data sets. The solid and dashed line denote the optimal separating hyperplane and the support hyperplanes, respectively.

Suppose we have a set of training data points $\mathbf{x}$ with their labeling information c_i as

$$\mathcal{S} = \{(\mathbf{x}_i, c_i) | \mathbf{x}_i \in \mathbb{R}^p, c_i \in \{1, -1\}\} \tag{4.3}$$

where $1, -1$ denotes two different classes, and the data point $\mathbf{x}_i$ is a p-dimensional

vector. Since any hyperplane in that space can be expressed by

$$\mathbf{w} \cdot \mathbf{x} - b = 0 \tag{4.4}$$

So the goal is to find a hyperplane $\mathbf{w}$ which can discriminate two different data sets, such that

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x} - b \leq -1, \quad \text{for all } c_i = -1 \\ \mathbf{w} \cdot \mathbf{x} - b \geq +1, \quad \text{for all } c_i = +1 \end{aligned} \tag{4.5}$$

This equation can be rewritten as

$$c_i(\mathbf{w} \cdot \mathbf{x} - b) \leq 1, \quad \text{for all } i \tag{4.6}$$

We can summarize the equations into the following optimization problem:

$$\begin{aligned} \text{choose } \mathbf{w}, b \text{ to maximize } ||\mathbf{w}|| \\ \text{which is subjected to } c_i(\mathbf{w} \cdot \mathbf{x} - b) \leq 1, \quad \text{for all } i \end{aligned} \tag{4.7}$$

The above equation is considered as the prime problem of the SVM. This problem can be solved by standard quadratic programming techniques and programs.

In our instrument classification system, a series of the training instrumental signal with the labeling information are used to calculate the optimal hyperplanes, which can be stored as the corresponding instrumental model in the system. The rest of the testing instrumental signal are then applied to scan which sites do they belong to according to the optimal hyperplanes.

4.3.3 k-Fold Cross Validation

The *cross validation* is a statistical way to partition the database into several parts, such that some of the parts are used as the training data and the others are used

to confirm and validate the performance. Among the various validation techniques, the k-fold cross validation is considered as one of the most often used methods. In k-fold cross validation, the original dataset is first partitioned into k subsets. Of the k subsets, a single subset is employed as the validation data for testing the performance, while the remaining $k-1$ subsets are used as the training data. The process is repeated k times, with each of the k subsets used exactly once as the validation data. We can then average the k performance estimation to derive a single estimation output. The advantage of this approach is that although the partition process is random, it ensures that all observations are used for both training and validation. $k = 5, 10$ both are the common used selections. In our instrument classification system, since the data sizes are considerably small, we use the 5-fold cross validation in both of the individual and instrument family classification tasks.

4.4 Simulation Results

Before performing the experiments, a data preprocessing is conducted to segment the instrumental signal into the isolated notes. The simulation results can be divided into two parts: the instrument family classification results and the individual instrument classification results. In both parts, the recognition processes are made according to the taxonomy illustrated in Figure 4.1.

4.4.1 Data Preprocessing

The experimental data used in this simulation are collected from the Musical Instrumental Samples[1] (MIS) offered by the Electronic Music Studios, University of Iowa.

[1] http://theremin.music.uiowa.edu/MIS.html

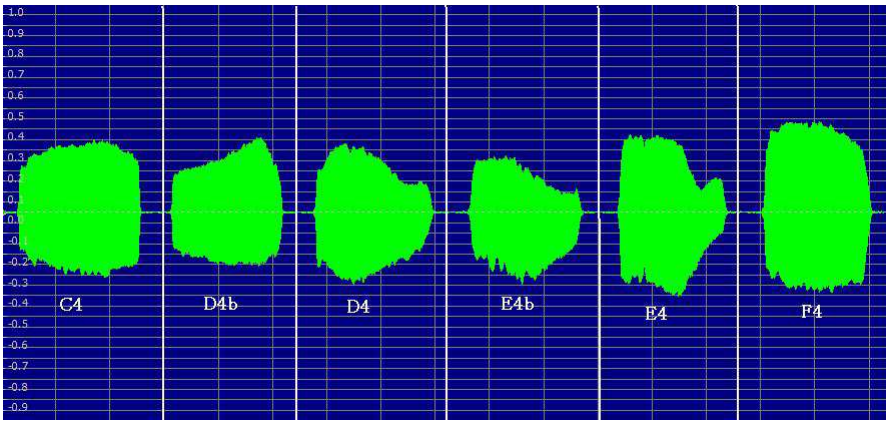


Figure 4.4: Example of segmenting the instrumental signal stream into individual notes.

In the MIS, all samples are in mono, 16 bit, 44.1 kHz, AIFF format. The exception is the piano, which is recorded in stereo. The database we collected consists of twenty various instruments, where some of them are both recorded with and without vibrato. The instrumental recordings offered by the MIS are with various volume of the notes, like *pp*, *mf* and *ff*. We only choose the *mf* (stands for *mezzo-forte*) volumes of the notes, since they are considered to be the moderately loud volumes.

Since most of the instrumental recording files in the MIS contain about one octave of the notes (i.e., totally twelve semi-tones), the segmentation and labeling process is conducted by hand, in order to verify the recognition performance in terms of the isolated individual notes. Figure 4.4 illustrates the segmentation and labeling processes. Therefore, after the data preprocessing, the database can be identified with 24 different categories, each with various number of notes and the pitch ranges. Table 4.1 lists the detail information of the monophonic database used in this chapter. Overall, the database consists of 973 various instrumental isolated notes.

4.4.2 Instrument Family Classification Results

Musical instruments form a hierarchical structure, which comprises various instrument families, as depicted in Figure 4.1. In some applications, classification down to the level of instrument families is sufficient for practical needs. For instance, a user may only query the music-searching system by the word "string" instead of the exact instrument such like the cell and violin. The accuracy of the instrument family classification is defined by

$$\text{Accuracy} = \frac{\#\text{of testing data belonging to the same family as query}}{\#\text{of testing data}} \tag{4.8}$$

Table 4.2 lists the resulting confusion matrix of the instrument family classification experiment. As can be shown, the overall recognition accuracy is 97.03%. According to the result, it seems that the recognition performance to the level of instrument families is enough for applying to the other content-based information retrieval applications.

4.4.3 Individual Instrument Classification Results

Similar to the previous one, the accuracy of the individual instrument classification is defined by

$$\text{Accuracy} = \frac{\#\text{of testing data belonging to the exact same instrument as query}}{\#\text{of testing data}} \tag{4.9}$$

Table 4.3 and 4.4 list the left and right half part of the confusion matrix of the individual instrument classification experiment, respectively. It appears that the recognition accuracy of the piano is the most highest one comparing to the other instruments. This may due to the fact that it is the unique instrument inside the

percussion family, and possesses different sound producing mechanism with other instruments. The other notable point is that since we separate the instrument played with and without vibration into different categories, the recognition performance degrades when the instrument has these two types of played modes.

4.5 Discussion

The aim of the study in this chapter is to verify the capability of automatically classifying the monophonic instrumental signal using various low-level features. The design of our classification system is similar to the one proposed by Deng *et al.* [53]. However, the experimental result shows that our recognition rates are higher than theirs in both of the individual and family classification cases. One possible reason is that we segment the continuous instrumental recordings, which contains a full musical scale, into a set of isolated notes, and thus resulting in the increase of the recognition rates. In addition, only the recordings with *mf* velocity are selected to construct the database. This may also cause the offset in the recognition results.

Table 4.1: List of the musical instrumental samples from *Electronic Music Studio* used in the experiment.

No.	Instrument	Family	Lowest	Highest	Size
01	Flute (without vib.)	Woodwind	B3	C7	38
02	Flute (with vib.)	Woodwind	B3	#C7	39
03	Alto Flute (with vib.)	Woodwind	G3	G6	37
04	Bass Flute (with vib.)	Woodwind	C3	#A5	35
05	Oboe	Woodwind	#A3	#G6	35
06	Eb Clarinet	Woodwind	G3	#F6	42
07	Bb Clarinet	Woodwind	D3	#C7	47
08	Bass Clarinet	Woodwind	#C2	B5	46
09	Bassoon (with vib.)	Woodwind	#A1	D5	41
10	Soprano Sax. (without vib.)	Woodwind	#G3	#D6	32
11	Soprano Sax. (with vib.)	Woodwind	#G3	#D6	32
12	Alto Sax. (without vib.)	Woodwind	#C3	#G5	32
13	Alto Sax. (with vib.)	Woodwind	#C3	#G5	32
14	French Horn	Brass	#A1	F5	44
15	Bb Trumpet (without vib.)	Brass	E3	D6	35
16	Bb Trumpet (with vib.)	Brass	E3	D6	35
17	Tenor Trombone	Brass	E3	C5	33
18	Bass Trombone	Brass	#C1	G4	43
19	Tuba	Brass	C1	C4	37
20	Violin (arco)	String	G3	#C7	43
21	Viola (arco)	String	C3	G6	44
22	Cello (arco)	String	C2	A5	46
23	Doublebass (arco)	String	E1	G4	40
24	Piano	Keyboard	B0	B7	85
Total Size:					**973**

Table 4.2: Confusion matrix of the instrument family classification results.

Instruments	Woodwind	Brass	String	Percussion
Woodwind	**97.0**	3.3	0.7	0
Brass	1.5	**95.3**	2.0	1.2
String	2.6	1.2	**96.2**	0
Percussion	0	0.4	0	**99.6**

Table 4.3: Confusion matrix of the individual instrument classification results (left-half part).

#	01	02	03	04	05	06	07	08	09	10	11	12
01	**85.4**	5.9	1.2	0	0	0.4	4.3	0	0	0	0	0
02	3.1	**86.9**	2.3	0	0	1.9	1.2	0	0	0	0	0
03	1.2	0.8	**89.1**	8.1	0	0	0	0	0.4	0	0	0
04	1.3	0.9	5.1	**88.0**	0	0	0	0	0.4	0	0	0
05	0.4	0.4	0	0	**94.0**	0.9	0	0	0	0	0	0
06	1.4	2.1	0.4	0	0.4	**85.4**	7.9	0.4	0.4	0	0	0.4
07	1.9	1.3	0.3	0	0.6	6.7	**86.0**	0.3	0.3	0.3	0	0.6
08	0.7	0.3	0	0	0	0	0.7	**93.8**	0	0.7	0	1.6
09	0	0	0	0.7	0	0	0.4	0	**98.5**	0	0	0
10	0	0.5	0.9	0	0	0	0.9	0	0	**87.3**	4.7	0.5
11	1.4	0	0.9	0	0	0	0	0.5	0	1.9	**89.7**	0.5
12	0.5	0	0	0	0.5	0	0.9	2.8	0.5	0.5	0.5	**88.3**
13	0	0	1.4	0	0	0	0	0	0	0.5	0.5	1.4
14	1.7	0.3	0	0	0	0	0.3	0.7	0	0	0	1.0
15	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0.3	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0.4	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0
21	0.3	1.4	0.7	0	1.7	0	0	0.3	0	0	0	0
22	0	0	0	0.7	3.3	0	0.3	0	0	0	0	0
23	0	0.4	0	0.8	0	0	0	0.8	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0

Table 4.4: Confusion matrix of the individual instrument classification results (right-half part).

#	13	14	15	16	17	18	19	20	21	22	23	24
01	0	0.4	0	0	0	0	0	0	2.0	0.4	0	0
02	0	0.8	0	0	0.4	0	0	0	2.7	0.8	0	0
03	0	0	0	0	0	0	0	0	0.4	0	0	0
04	0	0	0	0	0	0	0	0	0.9	0	3.4	0
05	0	0	0	0	0.4	0	0	0	1.3	2.6	0	0
06	0	0	0	0	0.4	0	0	0	0.7	0	0.4	0
07	0	0	0	0	0	0	0	0.3	0.6	0.3	0	0.3
08	0	0	0	0	0	0	0	0	1.6	0.3	0.3	0
09	0	0	0	0	0.4	0	0	0	0	0	0	0
10	2.3	1.4	0	0	0	0	0	0	1.4	0	0	0
11	2.8	0.9	0	0	0	0	0	0	1.4	0	0	0
12	1.4	3.8	0	0	0	0	0	0	0.5	0	0	0
13	**89.7**	2.3	1.4	0	0	0	0	0	2.8	0	0	0
14	0.3	**89.8**	0	0	0	4.8	0	0	1.0	0	0	0
15	0	0	**97.9**	2.1	0	0	0	0	0	0	0	0
16	0	0	3.9	**96.1**	0	0	0	0	0	0	0	0
17	0	0	0	0	**98.2**	1.8	0	0	0	0	0	0
18	0	1.4	0	0	1.7	**96.5**	0	0	0	0	0	0
19	0	0	0	0	0	0	**97.6**	0	0	0	0	0.4
20	0	0	0	0.7	0	0	0	**99.3**	0	1.6	0	0
21	0	0.7	0.3	0	0	0	0	0	**94.5**	0	0	0
22	0	0	0	0	0	0	0.3	0	0.7	**94.1**	0.7	0
23	0.4	0	0	0	0	0	0.4	0	1.5	0.8	**95.1**	0
24	0	0.3	0	0	0	0	0	0	0	0	0	**99.7**

Chapter 5

Instrumentation Analysis of Polyphonic Music

5.1 Introduction and Related Work

In Chapter 4, we already discussed the way to classify the musical signal of monophonic isolated notes. Now in this chapter, we extend the problem into a polyphonic situation. The term *polyphonic* here is used to describe the property in a music clip with more than one instrument existed at the same time. For instance, a Beethoven's violin sonata, which is composed of violin and piano, can be regarded as a polyphonic piece.

Generally, identifying the instrument set in a polyphonic music clip is more challenging than in a monophonic clip. This is due to the complex nature from mixing different sources together. Several research works have been conducted to deal with this problem [55]. First of all, Eggink *et al.* [56] developed a system that can identify a predominant solo instrument in the presence of an accompanying keyboard or

orchestra using the harmonic features and GMM classifiers. It makes a strong assumption that the harmonic structure of predominant melody should stick out than other instruments. In addition, they did not aim at identifying all the instrument set existed in the music clip. Kitahara *et al.* [57] tried to decompose the polyphonic problem into three sub-problems: feature variations caused by sound mixtures, pitch dependency of timbres and the musical context. However, it assumes that the note data information is already known. Essid *et al.* [58] exploited a taxonomy of music ensembles by applying the hierarchical clustering technique. Eggink *et al.* proposed another instrument identification method using the missing feature approach [59].

The above work all aims to identify the instruments appeared in the audio file, but with some of their own constraints or limitations. Nevertheless, they cannot reveal the instrumentation cue written by composers. In music, the term *instrumentation* emphasized here is referred to the way a composer arrange each individual instrument employed in a composition. To illustrate, say in a Beethoven's violin sonata, it is possible for the violin to dominate the melody in the first half of the progression, while the piano dominates the rest. Some instruments are very likely to appear in only a few segments but not the whole song. Instruments are also expected to change their roles of representing dominance and accompaniment during the progression. Generally, the dominance of each instrument should be examined and analyzed by its melody line, volume, and even the note component. Here we only use the volume hearing by the listeners to approximate its dominance. The aim of this work is to roughly manifest this time-varying instrumentation information.

The rest of this chapter is organized as follows. First, in Section 5.3 we will show the block diagram of our proposed system and clearly illustrate the detail information about each block. The system is then evaluated on a Western classical

music database to test its performance in Section 5.4. Finally, the discussion about the system is drawn in Section 5.5.

5.2 Motivation and a Small Experiment

To move from the monophonic case to the polyphonic problem, our idea is to continually utilize the same low-level features instead of designing a new one, but establish a bridge to overcome the semantic gap. We constructed a small experiment to identify how the polyphonic nature make influence on the feature calculation procedure. The feature vector employed in this section is derived from the MPEG-7 Audio Descriptor. Following is the detail information about these feature descriptors [24].

1. **Harmonic Spectral Centroid**

   The *Harmonic Spectral Centroid* descriptor is the amplitude-weighted mean of the harmonic peaks of the spectrum. Like the following three descriptors, it performs a basic fundamental frequency estimation algorithm and then calculate the power of the relative harmonics. It has similar property with the other centroid descriptors, but applies only to the harmonic part of the signal.

2. **Harmonic Spectral Deviation**

   The *Harmonic Spectral Deviation* descriptor relates to the spectral deviation of the log-amplitude components from a global spectral envelope.

3. **Harmonic Spectral Spread**

   The *Harmonic Spectral Spread* describes the amplitude-weighted standard deviation of the harmonic power of the spectrum, normalized by the instantaneous Harmonic Spectral Centroid.

4. **Harmonic Spectral Variation**

   The *Harmonic Spectral Variation* descriptor indicates the normalized correlation between the amplitude of the harmonic power between two subsequent time-slices of the signal.

5. **Log Attack Time**

   The *Log Attack Time* descriptor characterizes the "attack" property of a sound (i.e., the time it takes for the signal amplitude to rise from silence to the maximum). This descriptor captures the difference between a sudden and a smooth sound.

6. **Temporal Centroid**

   The *Temporal Centroid* descriptor also indicates the envelop of a signal, representing where in time the energy of a signal is focused. This descriptor aims at distinguishing between a decaying and a sustain note, when the lengths and the attack time of the two notes are identical.

The music material used in this experiment is the song *Let It Be* performed by the *Beatles*. In order to fully control the instrument and the volume, we use a MIDI synthesis software called *Reason* to synthesize the wave files in terms of various different situations. The song is fragmented into four clips, each with the duration equals to thirty seconds. There are two tracks in the MIDI files, that are played on the piano and the lead instruments. As mentioned earlier, the clip is first applied to a framing process and calculate the descriptors. The descriptors calculated from frames are averaged to get a final scalar value.

Figure 5.1 shows the result of the experiment. The six sub-plots correspond to six features from the MPEG-7 Audio Descriptors that was discussed before. The

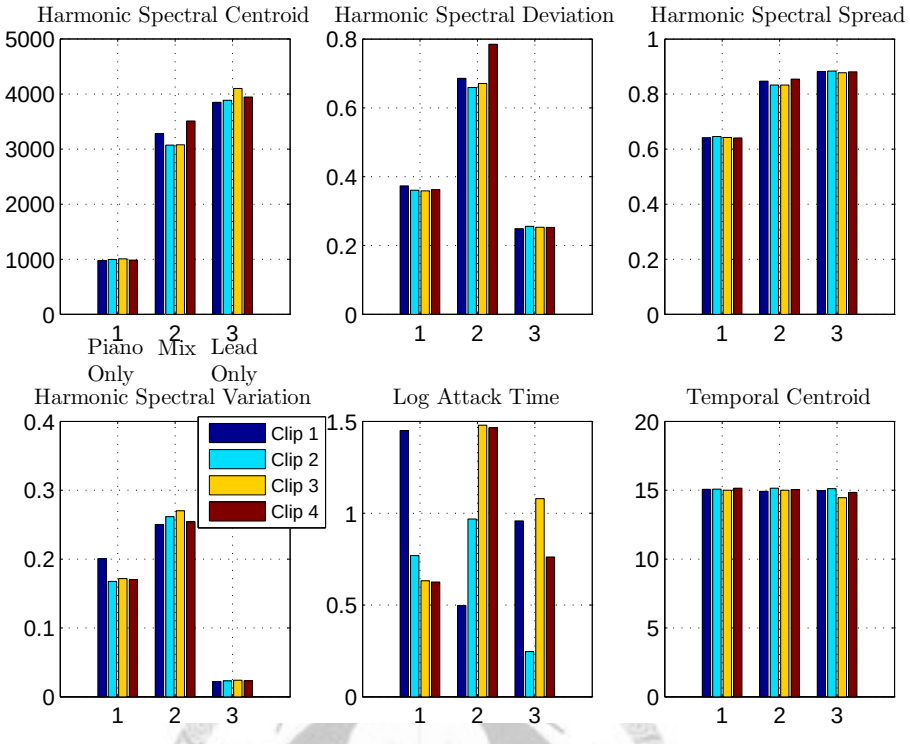


Figure 5.1: Experimental results of applying the MPEG-7 Audio Descriptors to polyphonic music.

four colors represent the four clips fragmented from the original song. The number in the x-axis (i.e., 1, 2 and 3) denote three different situations: only the piano track is presented, both piano and lead tracks are presented, and only the lead track is presented in the clip, respectively. It can be shown that the first four descriptors results in a great difference when the instrument set changes, and they are approximately independent to the changing of clips. For example, the Harmonic Spectral Centroid and the Harmonic Spectral Spread descriptors are larger when there is only the lead instrument presented, and are smaller when there is only the piano instrument presented. For clips mixed with two instruments, the values are in between. This is due to the fact that the descriptors somewhat have an averaging nature. For another, the Harmonic Spectral Deviation and the Harmonic Spectral Variation descriptors have the largest number when two tracks are mixed

Table 5.1: Corresponding velocity (volume) of piano and lead instruments in each song.

No.	Song01	Song02	Song03	Song04	Song05	Song06	Song07
Piano	120	120	110	100	90	80	70
Lead	N/A	70	70	70	70	70	70
No.	Song08	Song09	Song10	Song11	Song12	Song13	
Piano	70	70	70	70	70	N/A	
Lead	80	90	100	110	120	120	

and presented simultaneously. The experimental result reveals that the low-level features, such as the MPEG-7 Audio Descriptors, are able to capture the timbre variation even in the polyphonic music.

The purpose of the second experiment is to test how the volume of an instrument makes influence on the features. Similar to the first one, the song *Let It Be* is used in the experiment. We also apply *Reason* to synthesize thirteen songs, which correspond to various volume allocations in the MIDI file. Table 5.1 lists the detail information about the allocation of the velocities, which can be roughly regarded as the volume, of each song. The velocity of each instrument is a value that lies within 0 to 128. In the setting, the volume of the piano decreases when the song number increases. After the calculation of the features, the results are shown in Figure 5.2. It appears that the first four features (i.e., the Harmonic Spectral Centroid, Harmonic Spectral Deviation, Harmonic Spectral Spread and Harmonic Spectral Variation) tend to increase and decrease linearly when the volume allocation differs. Owing to this results, we believe that these basic features can still capture the instrument difference to some degree even in the polyphonic music. The following part of this section is to illustrate the proposed method of the instrumentation analysis system.

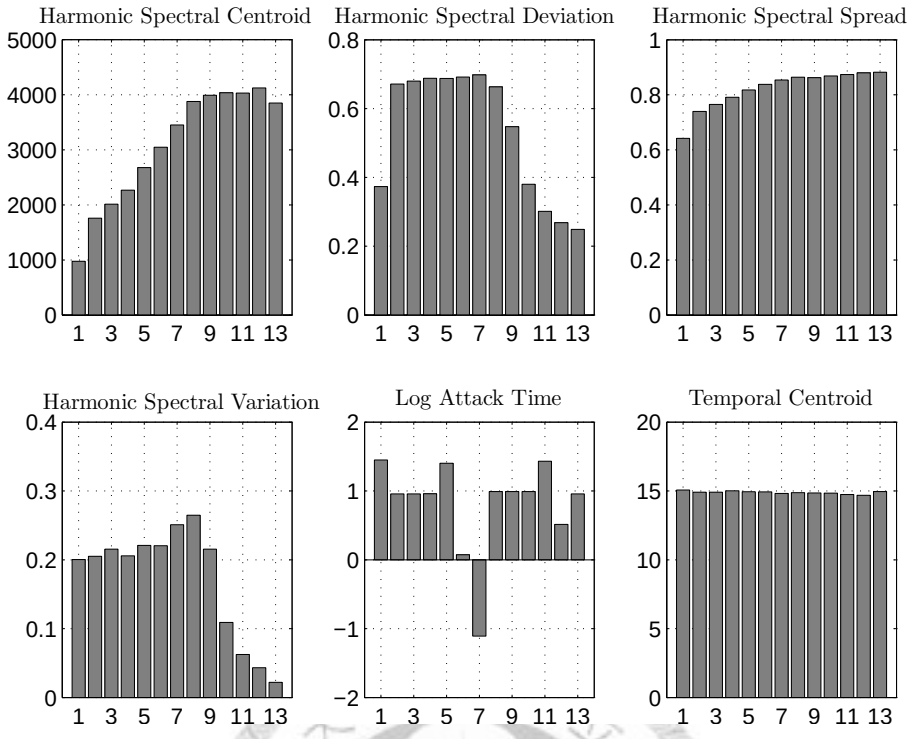


Figure 5.2: Experimental results of applying the MPEG-7 Audio Descriptors to polyphonic music.

5.3 Description of the Proposed System

The entire block diagram is shown in Figure 5.3. To begin with, the feature vectors and beat data of the input polyphonic music clip are extracted. After that an integration process is formed by averaging frames inside the same beat intervals. A fuzzy clustering algorithm is then applied to the integrated feature. The number of clusters should equal to the number of instruments. Finally, the cluster center and a few corresponding integrated features are used in the instrument identification process to determine the final instrument set.

5.3.1 Feature Extraction

In the first stage, we apply the low-level feature vector extraction to capture the basic signal characteristics from each frame. In order to simplify and unify the sys-

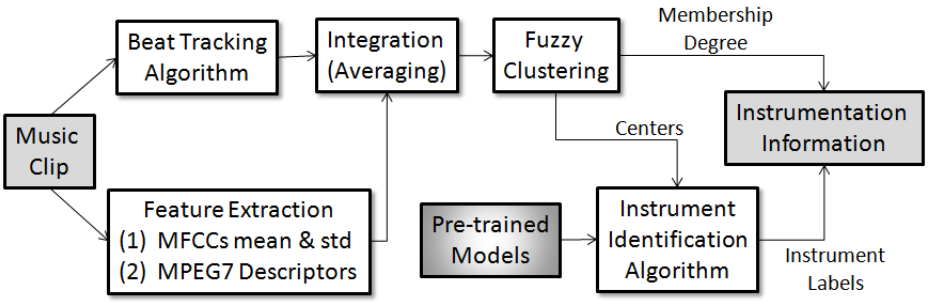


Figure 5.3: Block diagram of the proposed instrumentation analysis system.

Table 5.2: Detail of the feature vector used in this system.

#	**MFCC Features**
01 - 13	Mean of the first 13 MFCCs
14 - 26	Standard deviation of the first 13 MFCCs
#	**MPEG-7 Timbre Descriptors**
27	Harmonic Centroid Descriptor
28	Harmonic Deviation Descriptor
29	Harmonic Spread Descriptor
30	Harmonic Variation Descriptor
31	Spectral Centroid Descriptor
32	Temporal Centroid Descriptor
33	Log-Attack-Time Descriptor

tem, we select two low-level feature sets recommended in [53]. The study conducted several feature selection criteria to rank the features in the instrument recognition task. These two feature sets are the MPEG-7-based descriptors [60] and the MFCCs. Musical instrument timbre descriptors in MPEG-7 standard aim at describing perceptual features of instrument sounds. Collectively, they form a 33-dimensional feature vector of each frame. Table 5.2 lists the detail information about our feature vector. The input audio file is first converted to mono if needed, and then downsample to 16000 Hz to enhance the processing speed. After that, a hamming window with half-overlapped frame is applied to extract the feature vectors.

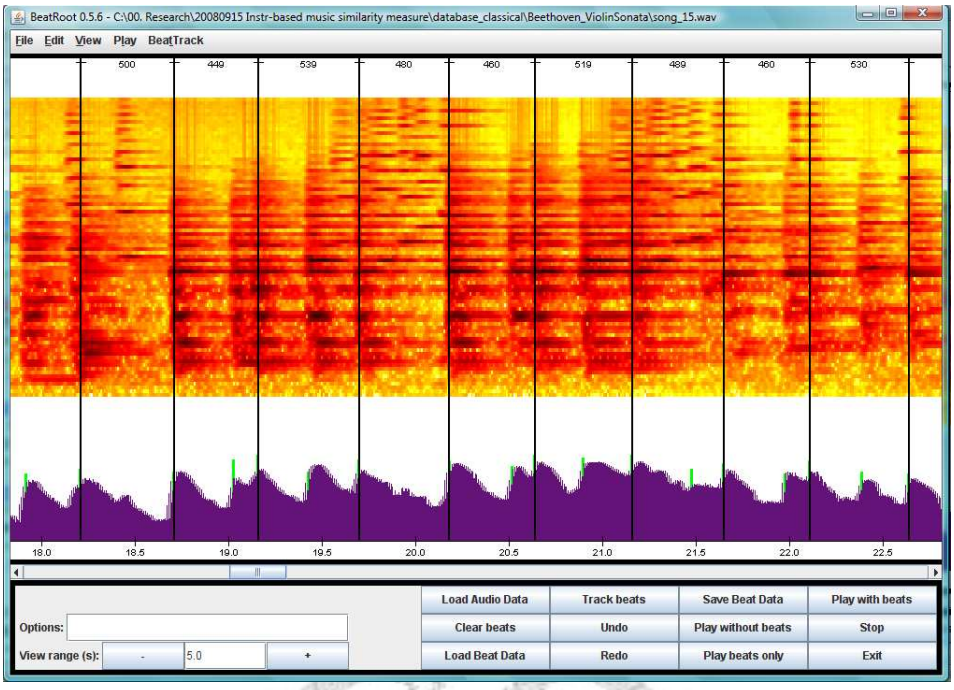


Figure 5.4: A demonstration of the *Beatroot* graphical user interface.

5.3.2 Beat Tracking and Feature Integration

So far, the feature vector obtained from each frame can only reveal the spectral information in a considerably short period, which is considered to be stationary. The integration of these fragmentary vectors needs to be accomplished according to the note data messages inside a music clip. However, since that calculating the exact onset time and duration of each note is still fairly challenging today, we adopt the beat-synchronous integration scheme suggested in [61] instead. The idea is, instead of integrating the feature vector inside a note, the integration process is constructed inside every beat interval. *BeatRoot* developed in [62] is used to perform the beat-tracking algorithm to the input signal. Figure 5.4 display a demonstration of the GUI of Beatroot. It estimates the beginning and the ending time of each beat and give an acceptable result for our further processing. Generally, a music clip tends to have only one or two notes appearing inside a beat time. The integration process

on beat interval could approximate the case on every exact note. Other minor exceptions can be treated as the outliers and be removed by a smoothing filter in the final stage.

Let $\mathbf{v}_{k,1}$, $\mathbf{v}_{k,N_k}$ denote the feature vectors calculated from the first and the last frame in the kth beat, respectively. Then the integration process can be done by averaging N_k vectors,

$$\mathbf{s}_k = \sum_{t=1}^{N_k} \frac{1}{N_k} \mathbf{v}_{k,t}, \tag{5.1}$$

where $\mathbf{s}_k$ denotes the beat-synchronous integrated vector in the kth beat.

5.3.3 Fuzzy Clustering

One of the major difficulty in instrument identification of polyphonic music is the timbre mismatch between the training and testing instruments. For instance, the violin used in the training process could not resemble the one in the testing music. Due to this reason, directly applying the beat-synchronous feature to supervised classifiers in a frame-by-frame manner would not perform well. On the other hand, we exploit the temporal continuity property in the instrumentation to solve this problem. That is, in most cases the instrument tends to be arranged consistently and the timbre of each specific instrument should not have a large change. We thus apply the fuzzy clustering technique to the integrated vectors of the entire music clip.

The fuzzy c-means clustering (FCM) algorithm attempts to partition a finite collection of elements $\mathbf{s}$ into a collection of c fuzzy clusters with respect to minimizing the following objective function Q:

$$Q = \sum_{i=1}^{c} \sum_{k} u_{ik}^{m} ||\mathbf{s}_k - \mathbf{c}_i||^2, \tag{5.2}$$

where m is a fuzzification coefficient ($m = 2$ in this paper). We use $\mathbf{c}$ and u_{ik} to denote the resulting cluster centers and the membership function. They can be obtained by iteratively repeating the following equations:

$$\mathbf{c}_i(t) = \frac{\sum_k u_{ik}^m(t)\mathbf{s}_k}{\sum_k u_{ik}^m(t)}, \tag{5.3}$$

and

$$u_{ik}(t+1) = \frac{1}{\sum_{j=1}^{c} (\frac{||\mathbf{s}_k - \mathbf{c}_i(t)||}{||\mathbf{s}_k - \mathbf{c}_j(t)||})^{2/(m-1)}} \tag{5.4}$$

Details of the algorithm can be found in [63]. Unlike hard k-means clustering, FCM gives the membership function as soft labeling, which can be regarded as the degree of dominance for a particular instrument. Experimental result shows that the volume of the instrument would directly make influence on the membership function output. Since estimating the number of instruments is beyond the scope of our work, we manually fed the correct number c into the system.

5.3.4 Instrument Identification

In instrument identification process, we use the support vector machine (SVM) as a supervised learning algorithm to accomplish the recognition task. The detail introduction of the SVM classifier can be found in Section 4.3.2.

After the integration process is finished, the remaining work is to identify the correct instrument represented by each cluster. Before the identification process, we use the SVM to build pre-trained models [64]. The training data is collected from different solo recordings, in order to consider the timbre variation between different music clips. Since that directly classifying the cluster centers using SVMs sometimes gives unfavorable results, we use an alternative method. A membership degree

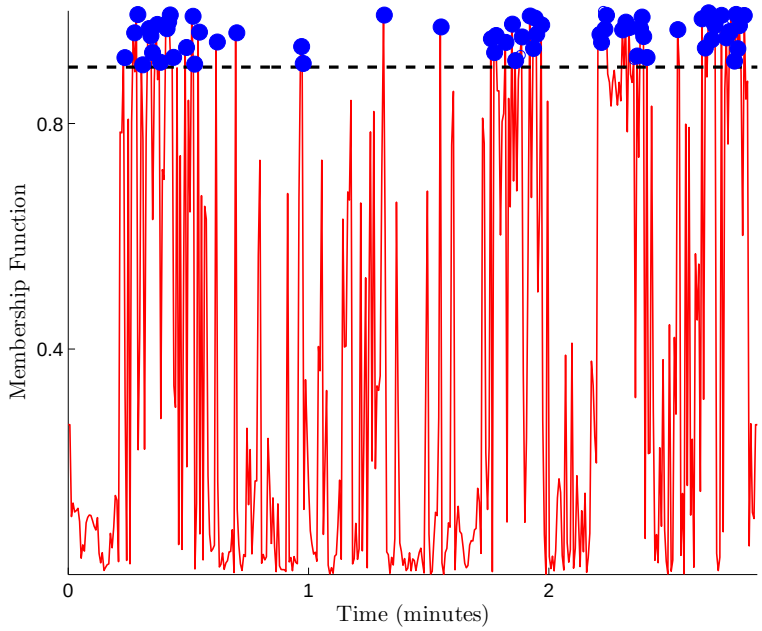


Figure 5.5: Instrument identification illustration: Selection of integrated vectors with their membership function exceeding the threshold.

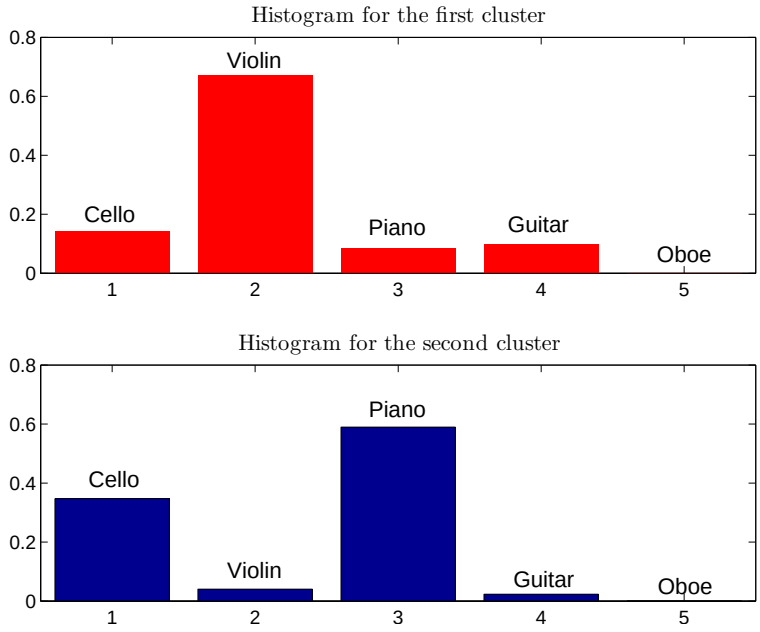


Figure 5.6: Instrument identification illustration: Calculating the instrument labeling histogram of selected integrated vectors using pre-trained SVM models for each cluster.

threshold T (0.9 in this paper) is set. For every cluster, integrated vectors with their membership function values higher than the threshold are grouped together and individually applied to SVMs to obtain the classification labeling result l_i of those integrated vectors.

$$l_i = \text{SVM}\{\mathbf{x}_k | u_{ik} > T\} \tag{5.5}$$

The result is then used to calculate the instrument labeling histogram. Figure 5.5 and 5.6 show an example of the integrated vectors selection and the histogram calculation result.

Let $H_{i,j}$ denote the jth instrument labeling count in the ith cluster derived from l_i, the labeling probability $P_{i,j}$ is generated by averaging the count $H_{i,j}$ within the same cluster as follows,

$$P_{i,j} = \frac{H_{i,j}}{\sum_{j=1}^{N_j} H_{i,j}} \tag{5.6}$$

The identification process is done by selecting the largest probability in $P_{i,j}$,

$$L_i = \arg\max_{i,j} P_{i,j}, \tag{5.7}$$

where L_i represents the identification result of the ith cluster. After the first step, the jth instrument is marked as already used in the histogram table. The system will continue to find the largest probability that exists in unused instruments and so on, until all clusters are labeled. For example, in Figure 5.6 cluster 1 will first be labeled as violin, and then cluster 2 will be labeled as piano. Finally, the labeled instrument set L_i and the membership function u_{ik} are treated as the instrumentation information output. This method is designed to solve the timbre mismatch problem. Since that the fuzzy clustering step is unsupervised, integrated vectors will automatically be clustered together with respect to different instruments, due

Table 5.3: Recognition rates for different instrument combinations in the Western classical music. Note that string is regarded as a combination of violin and cello here.

Violin Sonata			Cello Sonata
Violin	82.93%	Cello	85.71%
Piano	85.37%	Piano	90.48%
Piano Trio			Oboe Concerto
Piano	96.67%	Oboe	83.33%
Violin	81.11%	String	66.67%
Cello	94.44%	**Average**	**85.19%**

to its temporal continuity property. Classifiers are applied to the clustering result in the last stage.

5.4 Simulation Results

5.4.1 Experiment Setup

The evaluation of this work can be divided into two parts. First, the instrument identification process gives an estimation of the instrument set. We can calculate the averaging recognition rate by testing a set of duo and trio songs. For another, the instrumentation analysis result output a time-varying distribution related to each instrument. In this part we select two famous classical music clips to demonstrate our simulation results.

5.4.2 Instrument Identification Result

In this experiment, five common instrument models are trained by the SVM using clean solo recordings beforehand. They are cello, violin, piano, guitar and oboe.

Table 5.4: Number of training instrument models and average recognition rate comparing to other works.

	Kitahara *et al.* [65]	Essid *et al.* [58]	**Our works**
Number of models	4	12	**5**
Recognition rate	83.30%	53.00%	**85.19%**

The average length of training data for each model is about 50 minutes. To evaluate the identification performance, instead of using the MIDI-based synthesized files, we select four regular musical forms in real-world Western classical music as the testing data. Each of them is composed of different instrument sets. The database consists of 200 music clips, and the overall duration of the music clips is about 10 hours. The recognition accuracy of instrument i is defined by

$$\text{Accuracy}_i = \frac{\#\text{ of clips correctly identified as } i}{\#\text{ of testing music clips}} \tag{5.8}$$

The result is listed in Table 5.3. It results in an 85.19% recognition rate in average, which is essentially comparable to other relative works in terms of the training model size [65], [58]. Table 5.4 lists a comparison of the averaging recognition rates with other relative works. One should notice that the larger number of training models leads to the lower average recognition rate. In addition, the training and testing data used in these works are not exactly the same.

5.4.3 Instrumentation Analysis Result

Membership function output from the fuzzy clustering algorithm combined with the instrumental labels are considered as the dominance of the instrument played in a music clip. We select two well-known Western classical music pieces to demonstrate the result. They are a violin sonata composed by Beethoven, and a piano trio

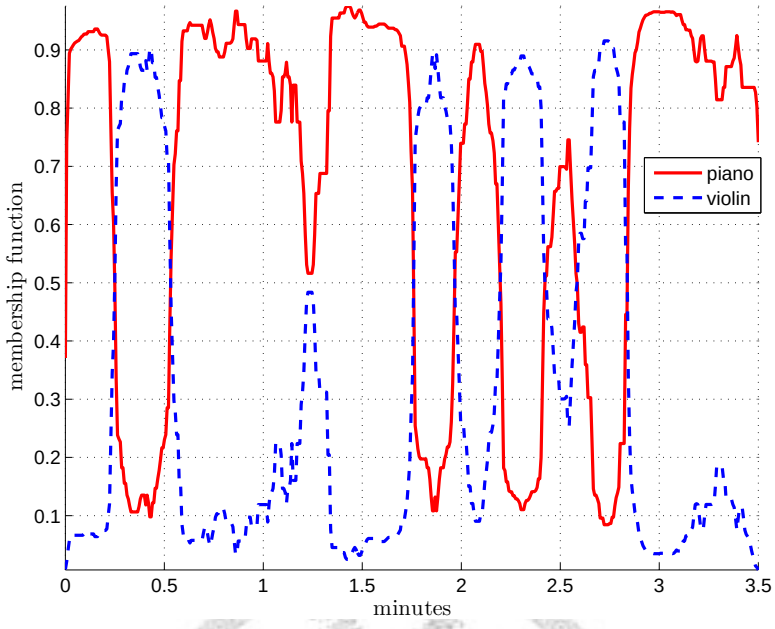


Figure 5.7: Simulation results of "Beethoven: Violin Sonata Spring mov.4". Only the 3.5 minutes in beginning is selected.

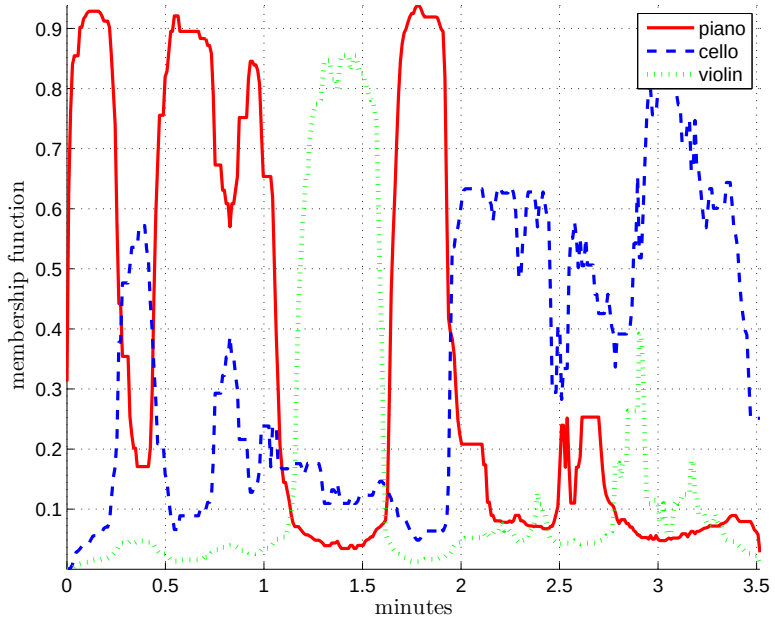


Figure 5.8: Simulation results of "Brahms: Piano Trio No.4 mov.3". Only the 3.5 minutes in beginning is selected.

composed by Brahms. Figure 5.7 and Figure 5.8 show the results of these duo and trio pieces, respectively. By referring to the music scores and recordings, it can be shown that the system output a reasonable estimation.

5.5 Discussion

In this chapter, an instrumentation analysis algorithm for polyphonic music is proposed. The system can automatically identify the instrument set without knowing any of the note data information, such like the pitch and onset timing. As mentioned earlier, the system takes considerably less computation time. It only requires a beat-tracking algorithm with fuzzy clustering technique. We examined it with five common instrument models, and find that the result is comparable to other relative works. Moreover, it can roughly sketch the dominance of each instrument during the music progression. We believe this time-varying result can be used as a mid-level feature to improve the performance of music recommendation systems. To be more specific, the systems may recommend a list of songs which is similar in their instrumentation information as a new option, instead of using the metadata (e.g., the genre and artist).

For future work, we would like to increase the number of training instruments. The system will be modified to further accommodate to drum and human voice, which are very common in recent popular music. The inharmonic nature of these signals needs to be handled by extra algorithms.

Chapter 6

An Instrumentation-Based Music Similarity Measure System

6.1 Introduction and Related Work

The advent of the audio compression techniques and the booming Internet technology has changed the way people listens to and discover new music. Owing to the rapid growth in the digital music collection size, the conventional way of discovering and recommending music, such as the radio broadcasting, is no longer sufficient for the immense music files nowadays. As a result, a computer-based and personalized music recommendation system is highly expected [2].

One of the most crucial steps in implementing such system is calculating the similarity between songs. Several algorithms have been proposed and evaluated by researchers toward this topic [10], [12], [14], [16], [17]. However, most of the existing systems rely mainly on calculating the statistical model of the low-level features, such as the Mel-frequency cepstral coefficients (MFCCs), inside a music

clip. These low-level features, in spite of their highly efficient and uniform nature, could not access to any of the musical content (e.g., the melody, chord progression or instrumentation). This shortage has been investigated in [61], [66]. In [66], the authors incorporated the chord progression estimation into the feature extraction step to boost the accuracy of musical emotion classification.

In music, the term *instrumentation* is referred to the way a composer arrange the individual instrument employed in a composition. In our previous work [6], we proposed an algorithm to analyze the musical signal and derive its instrumentation estimation. Unlike the existing algorithms, it not only gives the identification result, but also output a time-varying information of each instrument. This information can be further employed to design several mid-level features, which are well correlated with the musical content, to enhance the similarity calculation procedure. Moreover, by combining these mid-level features, our similarity calculation result can be specifically oriented toward the instrumentation cue, except for other content. This property should be fairly useful for many music information retrieval tasks.

The rest of this chapter is organized as follows. In Section 6.2 we briefly introduce the structure and functionality of the instrumentation analysis system. Then in Section 6.3 we illustrate the way to utilize the instrumentation information as a mid-level feature to construct a similarity measure system. This system is evaluated by an objective test in Section 6.4. Finally, the conclusions and future work are drawn in Section 6.5.

6.2 Instrumentation Analysis System

The purpose of our instrumentation analysis system is to identify the instrument set inside a music piece, and then draw a rough sketch of their time-varying dominance as an additional information. The detail algorithm can be found in Chapter 5. First of all, the low-level feature vector, which combines the mean and variance of the MFCCs and MPEG-7 Audio Descriptors are extracted. We also apply *BeatRoot* to extract the beat data of the input music piece. The feature vector is then integrated by averaging frames inside the same beat interval. This integration process leads to a more compact and content-based feature representation. Instead of classifying each integrated vectors independently, an unsupervised fuzzy c-means algorithm is applied. The number of clusters c is equal to the number of instruments appearing in the music piece. This approach employs the time coherence of instruments inside a music piece (i.e., the consistency of the instrument set throughout the music), and also aims at solving the timbre mismatch problem between the training and testing data. Then the cluster centers with a few corresponding integrated vectors are used in the instrument identification process, which consists of a set of SVMs and the pre-trained models, to identify the final instrument set. The membership function related to a cluster is regarded as the time-varying dominance of that instrument. Finally, the identification instruments with their dominance are considered as the output of the system.

6.3 Proposed Similarity Measure System

Figure 6.1 shows the block diagram of the proposed instrumentation-based music similarity measure system. For a music database that consists of N songs, first

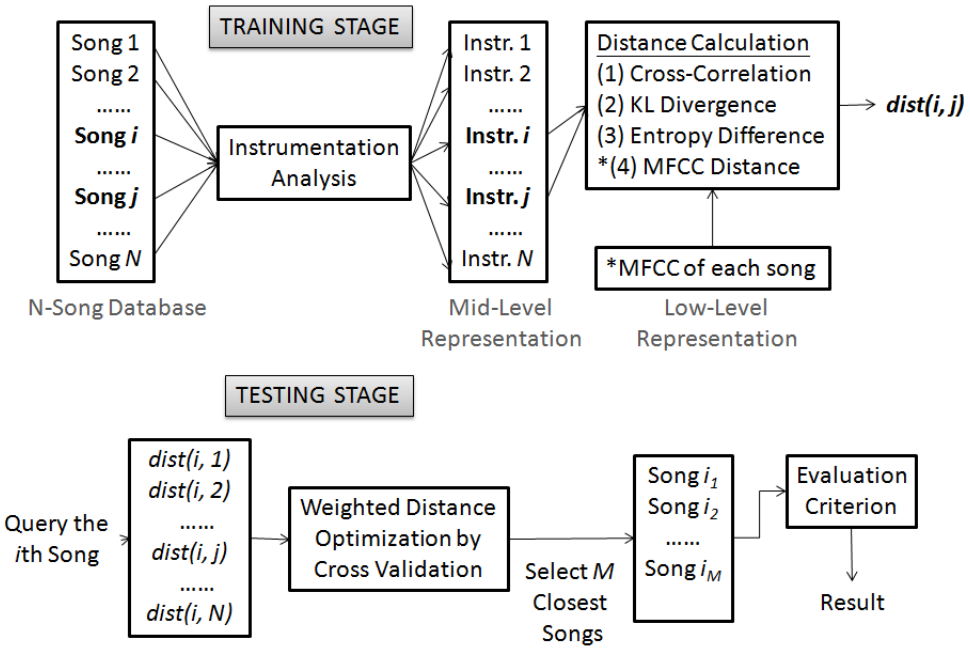


Figure 6.1: Block diagram of the proposed instrumentation-based music similarity measure system.

we apply the instrumentation analysis algorithm to extract the instrumentation information of all the songs. The ith instrumentation information is denoted as $I_i(m_i, n_i)$, where m_i, n_i are the instrument and the beat interval index, respectively. After obtaining these mid-level representations, we design a distance calculation step to calculate the distance vector of all the songs in pairs. This design mainly focuses on the song pairs with the same instrument set (e.g., two violin sonatas). The distance calculation between different instrument sets is beyond the scope of this paper. The distance vector of I_i and I_j, denoted as $\mathbf{dist}_{i,j}$, can be expressed as a four-dimensional vector

$$\mathbf{dist}_{i,j} = [\mathrm{CC}, \mathrm{KL}, \mathrm{ED}, \mathrm{MD}]^T \tag{6.1}$$

These four values is calculated as follows.

6.3.1 Normalized Cross-Correlation

The normalized cross-correlation of instrument m of I_i and I_j is

$$\mathrm{CC}_m(\tau) = \frac{1}{\min(L_i, L_j)} \sum_{n=0}^{\infty} I_i(m,n) \cdot I_j(m, n+\tau) \tag{6.2}$$

where L_i and L_j denote the data length (time duration) of I_i and I_j. By averaging all instruments with their maximal values and taking the inverse, we can obtain the distance

$$\mathrm{CC} = (\frac{1}{M} \sum_{m} \max_{\tau}(\mathrm{CC}_m(\tau)))^{-1} \tag{6.3}$$

where M is the number of instruments. This coefficient is used to measure the similarity of two waveforms as a function of a time-lag applied to one of them.

6.3.2 Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence is used to measure the difference between two instrument distributions. Let $P_i(m)$, $P_j(m)$ denote the instrument distributions of I_i and I_j by calculating the mean membership value according to time, such as

$$P_i(m) = \frac{1}{L_i} \sum_{n_i} I_i(m, n_i) \tag{6.4}$$

The KL divergence from $P_j(m)$ to $P_i(m)$ is calculated as

$$D_{\mathrm{KL}}(P_i||P_j) = \sum_{m=1}^{M} P_i(m) \log \frac{P_i(m)}{P_j(m)} \tag{6.5}$$

Owing to the asymmetric property of the divergence, we made it become symmetric by summing up from two directions:

$$\mathrm{KL} = D_{\mathrm{KL}}(P_i||P_j) + D_{\mathrm{KL}}(P_j||P_i) \tag{6.6}$$

6.3.3 Entropy Difference

Let $\text{hist}_{i,m}(X)$ denote the histogram of the mth instrument in I_i, the entropy $H_{i,m}(X)$ can be calculated by

$$H_{i,m}(X) = -\sum_{k} \text{hist}_{i,m}(x_k) \log_2 \text{hist}_{i,m}(x_k) \tag{6.7}$$

We can then calculate the absolute difference between $H_{i,m}$ and $H_{j,m}$ and average the difference with respect to all the instruments to obtain the distance

$$\text{ED} = \frac{1}{M} \sum_{m=1}^{M} |H_{i,m} - H_{j,m}| \tag{6.8}$$

The entropy of an instrumentation function can be regarded as an estimation of the changing speed of the dominance and accompaniment instruments inside a music piece.

6.3.4 MFCC Distance

Unlike the above three features, the MFCC distance (MD) is directly derived from the low-level feature MFCCs. We apply *MA Toolbox* developed by Pampalk to calculate the distance [20]. MFCCs are first extracted from each song, and then applied to a k-means clustering step to obtain a statistical model. Finally, the Monte-Carlo sampling method is applied to approximate the likelihood function of the songs in pair as a distance measure MD.

6.3.5 Weighted Distance Optimization

In the testing stage, we apply a weighted average to the distance vector to obtain the final scalar distance $d_{i,j}$,

$$d_{i,j} = \mathbf{c} \cdot \mathbf{dist}_{i,j} \tag{6.9}$$

Table 6.1: Detail information of each subset inside the testing database.

Subset	Number of pieces	Class	Instruments
Violin Sonata	135	Duo	Violin & Piano
Cello Sonata	20	Duo	Cello & Piano
Piano Trio	47	Trio	Piano & Violin & Cello

where $\mathbf{c} = [c_1, c_2, c_3, c_4]$ and is subjected to

$$\sum_r c_r = 1, \quad 0 \leq c_r \leq 1, \quad \text{for all } r. \tag{6.10}$$

The way of choosing the coefficient $\mathbf{c}$ is based on a 10-fold cross validation in the experiment. That is to say, $\mathbf{c}$ is selected to maximize the accuracy while nine-tenth of the music pieces represent the training data. After $\mathbf{c}$ is determined, the rest one-tenth of the pieces are used to validate the performance. The process is then repeated ten times to average the result.

In our work, $\mathbf{c}$ is selected to maximize the accuracy while nine-tenth of the music pieces represents the training data. After $\mathbf{c}$ is determined, the rest one-tenth of the pieces is used to validate the performance. The process is then repeated ten times to average the result.

6.4 Simulation Results

In our instrumentation analysis system, the pre-trained model comprises five instruments: the violin, cello, piano, guitar and oboe. Due to its limitation to detect the drum pattern and human voice, we operate the system on a Western classical music database, instead of the popular songs. The database consists of three subsets: two duo classes and one trio class. Table 6.1 lists the detail information of them. Overall, the database contains 202 pieces, which are collected from the work

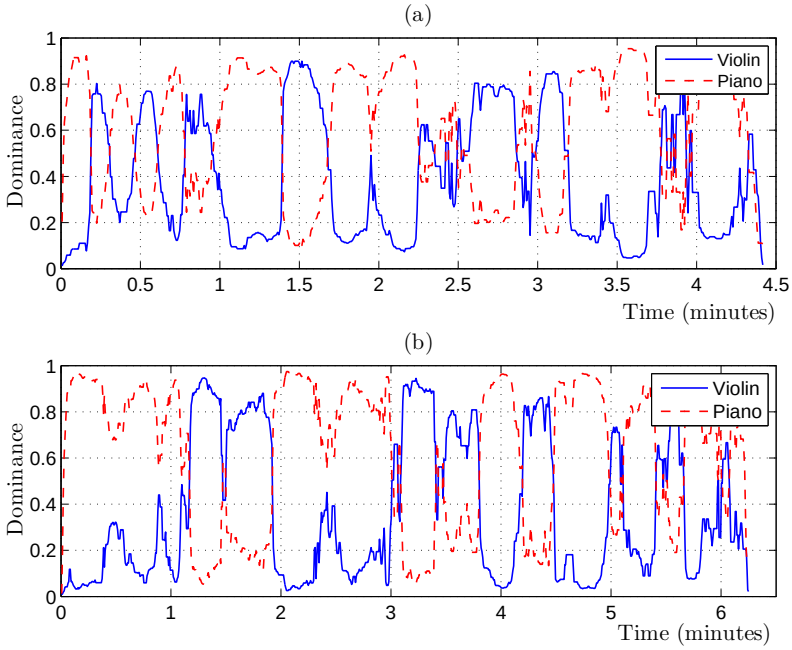


Figure 6.2: A query-and-hit example. The top plot is instrumentation estimation of the query song, Mozart violin sonata KV. 380, mov. 3. The bottom plot shows the estimation of the hit song, Mozart violin sonata KV. 296, mov. 1. In this example $\mathbf{c} = [0.73, 0.18, 0.04, 0.05]$.

composed by the Baroque, Classics, Romantic and Contemporary composers (e.g., Bach, Beethoven, Brahms, Chopin, and Prokofiev). All songs are converted to the 16000 Hz and mono wave files if needed.

To date, the way of choosing the proper evaluation criterion is still very challenging for music similarity applications. In this study, we adopt the composer classification criterion and measure the accuracy to evaluate the performance. According to music theory, the pieces composed by the same composer tend to share similar properties, such like the instrumentation and chord progression. Therefore, the classification performance is expected to be improved while we integrate these content-based information in the system.

For each input query song, first we compute its distances with all other songs inside the database, as described in the previous section, and then apply the k-nearest

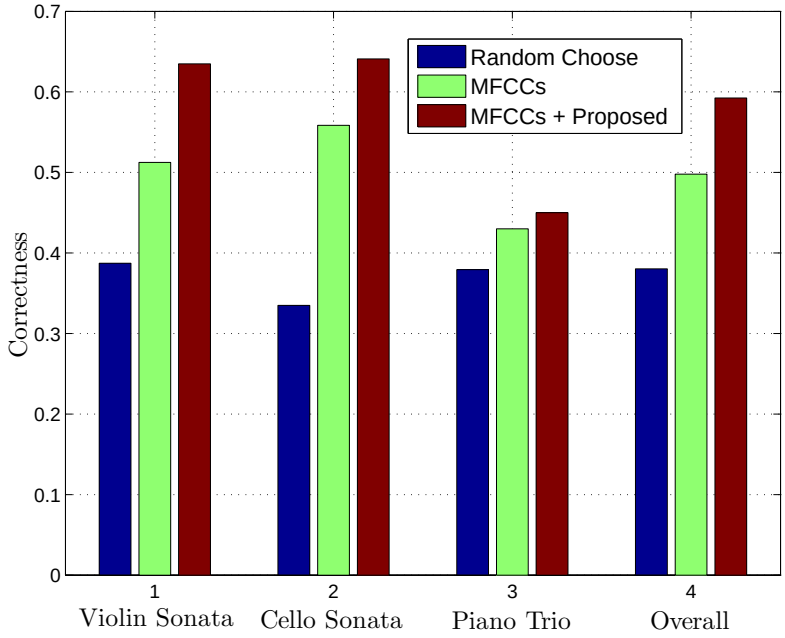


Figure 6.3: Evaluation result of the similarity measure system. It displays the accuracy of each subset inside the database using different feature schemes. Overall, by incorporating the proposed features the accuracy can be increased by nearly 10%.

neighbor (kNN) to select the k closest songs. Figure 6.2 shows an example of the successful query and hit songs with their instrumentation estimations. These songs are both the movements from the violin sonatas composed by Mozart. According to the plot, it appears that although having distinct durations, the two songs possess several similar properties in their instrumentation estimations, such like the distribution of the instrument and the changing speed of the dominance and accompaniment instruments. To quantitatively analyze the performance, the accuracy is defined by

$$\text{Accuracy} = \frac{\#\text{ of music with the same composer as query}}{k} \tag{6.11}$$

Three different approaches are compared in this experiment. The random-choosing approach is considered as the baseline. We also compare the cases of using the MFCCs only and the one integrated with the proposed features.

Figure 6.3 illustrates the accuracy of each individual subset and the overall performance in terms of different approaches. In all subsets, the classification accuracies tend to increase by incorporating the proposed mid-level features. It seems that the increase in duo classes are greater than the one in the trio class. One possible explanation is the degradation of the instrumentation estimation correctness when the number of instruments inside the piece increases. Generally, the integration results in an increase from 49.79% to 59.24% in overall classification performance in our experiment. The result shows the proposed features are fairly helpful since they can access the musical content.

6.5 Discussion

In this chapter, we present a music similarity measure system based on the instrumentation information instead of merely using the low-level feature. Three instrumentation-related mid-level features are designed and combined with the MFCCs together to improve the measure performance. This approach aims at overcoming the semantic gap between the signal characteristics and the musical content, which is considered as one of the most challenging tasks in content-based music information retrieval applications. We examine the system on a classical music database. The result shows that by incorporating the mid-level features, the composer classification accuracy can be further increased by approximately 10%.

For future work, there are two possible directions. First, we are still refining the instrumentation analysis system by detecting the drum patterns and human voices, in order to handling the real-world popular music. For another thing, except for the instrumentation, other information (e.g., melody and tempo) is also likely to

increase the performance in the similarity measure tasks.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The development of the feature extraction techniques in audio signal processing led us to infer that the music similarity measure system designed by such techniques can be well consistent with the human auditory system. The simulation results, as shown in Chapter 2, clearly supports this expectation. The result shows that the retrieved similar songs tend to share the same instrument arrangement or the genre categories with the original query song. One possible explanation is that the signal processing techniques utilized in this study, such like the MFCCs, could capture the characteristics of the individual timbre to some degree. Generally, the musical timbre has a strong influence on classifying the songs according to their genres and instrument arrangements.

Although the results seems to fulfill our expectation, this design still leaves room for improvement if we want to incorporate it into a personalized music recommendation system. That is, in such applications it is required to design a new mid-level

features that directly orients toward a specific musical content (e.g., the genre and instrumentation). This is the main topic in the following chapters. In Chapter 3, we extend the idea of the constant Q transform into the time-frequency plane, to examine the distributions of various instrumental signals. The experimental result shows that the time-frequency analysis using the CQT is more suitable for analyzing the instrumental signals when dealing with the pattern recognition problems.

Based on the findings from the analyzing results of the musical instrumental signals, we examine the usage of various feature sets to classifying the real-world instrumental pieces in terms of both the monophonic and polyphonic cases. Recognizing the instrumental trajectories in the polyphonic music is more challenging than the monophonic music. In this study, we continually utilize the low-level features instead of designing a brand new one, but with the temporal feature integration strategy and the fuzzy clustering techniques. In Chapter 5, a new instrumentation analysis system is proposed. The main difference of our proposed system and the related works is that we try to estimate the dominance of each existing instrument with respect to time. These time-varying information seems to accord with the volumes of the instrumental sounds, which is considerably consistent with the human perception.

Due to the success of estimating the instrumentation information, we further utilize these information as mid-level feature sources, and design a instrumentation-based music similarity measure system. In contrast with the system discussed in Chapter 2, the proposed system can be specifically oriented toward the instrument content, but not other musical contents. The performance has been evaluated by a classical music database with an objective test. The result shows that the incorporation of these newly designed mid-level features can enhance the composer

classification accuracies.

7.2 Future Work

This study has taken a step in the direction of designing a new mid-level feature extraction method and music similarity measure system. The study conducted several representative experiments based on the small-scale music databases. We acknowledge that although we tried to cover a wide range of different music styles, the results could not be generalized to all the possible real-world music pieces, owing to the limitation of the database sizes. In addition, the subjective nature of the music similarity seems to restrict the capability for quantitatively analyzing the performance of our systems. The approaches outlined in this study should be replicated in other large-scale music databases consisting the other real-world music pieces. To achieve this, further research may include the design of recognizing the percussion instruments, such as a drum set, which is considered to be non-harmonic. Moreover, in most of the popular music, the arrangers tend to use lots of the synthesized instruments, which are difficult to be recognized and classified correctly.

References

[1] K. Brandenburg and M. Bosi, "Overview of MPEG audio: Current and future standards for low-bit-rate audio coding," *Journal-Audio Engineering Society*, vol. 45, pp. 4–21, 1997.

[2] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[3] S. Douglas, "Music downloads reach record high," 2007, [online] available at http://www.investmentmarkets.co.uk.

[4] M. Castelluccio, "The Music Genome Project," *Strategic Finance*, vol. 88, no. 6, pp. 57–58, 2006.

[5] H. C. Chen and A. L. P. Chen, "A music recommendation system based on music and user grouping," *Journal of Intelligent Information Systems*, vol. 24, no. 2, pp. 113–132, 2005.

[6] S. C. Pei and N. T. Hsu, "Instrumentation analysis and identification of polyphonic music using beat-synchronous feature integration and fuzzy clustering," in *Proceedings of the ICASSP*, 2009, (accepted).

[7] S. C. Pei and N. T. Hsu, "A novel music similarity measure system based on instrumentation analysis," in *Proceedings of the ICME*, 2009, (accepted).

[8] C. S. Xu, N. C. Maddage, and X. Shao, "Automatic music classification and summarization," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 3, pp. 441–450, 2005.

[9] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[10] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *Proceedings of the ICME*, 2001, pp. 952–955.

[11] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[12] E. Pampalk, "Speeding up music similarity," in *Proceedings of the MIREX*, 2005.

[13] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *Proceedings of the ISMIR*, 2005.

[14] E. Pampalk, A. Rauber, and D. Merkl, "Content-based organization and visualization of music archives," in *Proceedings of the ACM International Conference on Multimedia*, 2002, pp. 570–579.

[15] E. Pampalk, S. Dixon, and G. Widmer, "Exploring music collections by browsing different views," *Computer Music Journal*, vol. 28, no. 2, pp. 49–62, 2004.

[16] J. J. Aucouturier and F. Pachet, "Music similarity measures: What's the use?," in *Proceedings of the ISMIR*, 2002.

[17] J. J. Aucouturier and F. Pachet, "Improving timbre similarity: How high is the sky?," *Journal of Negative Results in Speech and Audio Sciences*, vol. 1, no. 1, pp. 1–13, 2004.

[18] J. J. Aucouturier, F. Pachet, and M. Sandler, "The way it sounds: Timbre models for analysis and retrieval of music signals," *IEEE Transactions on Multimedia*, vol. 7, no. 6, pp. 1028–1035, 2005.

[19] G. Tzanetakis and P. Cook, "MARSYAS: A framework for audio analysis," *Organised Sound*, vol. 4, no. 3, pp. 169–175, 2000.

[20] E. Pampalk, "A Matlab toolbox to compute music similarity from audio," in *Proceedings of the ISMIR*, 2004, pp. 254–257.

[21] T. Kolenda, S. Sigurdsson, O. Winther, L. K. Hansen, and J. Larsen, "DTU: Toolbox," 2002, [online] available at http://isp.imm.dtu.dk/toolbox/.

[22] X. Huang and H. W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*, Prentice Hall PTR, 2001.

[23] J. S. Jang, "Audio Processing Toolbox," [online] available at http://www.cs.nthu.edu.tw/ jang.

[24] J. M. Martinez, "MPEG-7 overview," *ISO/IEC JTC1/SC29/WG11N5525*, 2003, [online] available at http://www.chiariglione.org/mpeg/standards/mpeg-7.

[25] S. Quackenbush and A. Lindsay, "Overview of MPEG-7 audio," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 725–729, 2001.

[26] J. J. Aucouturier and F. Pachet, "The influence of polyphony on the dynamical modelling of musical timbre," *Pattern Recognition Letters*, vol. 28, no. 5, pp. 654–661, 2007.

[27] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[28] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[29] J. S. Jang, "DCPR (Data Clustering and Pattern Recognition) Toolbox," [online] available at http://www.cs.nthu.edu.tw/ jang.

[30] S. Chretien and A. O. Hero III, "Kullback proximal algorithms for maximum-likelihood estimation," *IEEE Transactions on Information Theory*, vol. 46, no. 5, pp. 1800–1810, 2000.

[31] X. Zhou, Y. Fu, M. Liu, M. Hasegawa-Johnson, and T. S. Huang, "Robust analysis and weighting on MFCC components for speech recognition and speaker identification," in *Proceedings of the ICME*, 2007, pp. 188–191.

[32] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[33] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Journal of the American Society for Information Science*, vol. 41, no. 4, pp. 288–297, 1990.

[34] J. C. Brown, "Calculation of a constant Q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[35] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *The Journal of the Acoustical Society of America*, vol. 92, no. 5, pp. 2698–2701, 1992.

[36] C. N. dos Santos, S. L. Netto, L. W. R. Biscainho, and D. B. Graziosi, "A modified constant-Q transform for audio signals," in *Proceedings of the ICASSP*, 2004, vol. 2, pp. 469–472.

[37] D. B. Graziosi, C. N. dos Santos, S. L. Netto, and L. W. P. Biscainho, "A constant-Q spectral transformation with improved frequency response," in *Proceedings of the ISCAS*, 2004, vol. 5, pp. 544–547.

[38] F. C. C. B. Diniz, I. Kothe, L. W. P. Biscainho, and S. L. Netto, "A bounded-Q fast filter bank for audio signal analysis," in *Proceedings of the International Telecommunications Symposium*, 2006, pp. 1015–1019.

[39] F. C. Diniz, L. W. P. Biscainho, and S. L. Netto, "Practical design of filter banks for automatic music transcription," *International Symposium on Image and Signal Processing and Analysis*, pp. 81–85, 2007.

[40] O. Izmirli, "A hierarchical constant Q transform for partial tracking in musical signals," in *Proceedings of the 2nd COST G-6 Workshop on Digital Audio Effects*, 1999.

[41] W. J. Pielemeier, G. H. Wakefield, and M. H. Simoni, "Time-frequency analysis of musical signals," *Proceedings of the IEEE*, vol. 84, no. 9, pp. 1216–1230, 1996.

[42] M. Lawrence, "Simple music theory as it relates to signal processing," 2004, [online] available at http://cnx.org/content/m12461.

[43] Z. Duan, Y. Zhang, C. Zhang, and Z. Shi, "Unsupervised single-channel music source separation by average harmonic structure modeling," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 4, pp. 766–778, 2008.

[44] A. Eronen and A. Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *Proceedings of the ICASSP*, 2000, vol. 2, pp. 753–756.

[45] A. Eronen, "Comparison of features for musical instrument recognition," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.

[46] A. G. Krishna and T. V. Sreenivas, "Music instrument recognition: from isolated notes to solo phrases," in *Proceedings of the ICASSP*, 2004, vol. 4, pp. 265–268.

[47] S. Essid, G. Richard, and B. David, "Musical instrument recognition on solo performance," in *Proceedings of the EUSIPCO*, 2004, pp. 1289–1292.

[48] E. Benetos, M. Kotti, and C. Kotropoulos, "Musical instrument classification using non-negative matrix factorization algorithms and subset feature selection," in *Proceedings of the ICASSP*, 2006, vol. 5, pp. 221–224.

[49] S. Essid, G. Richard, and B. David, "Musical instrument recognition by pairwise classification strategies," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 4, pp. 1401–1412, 2006.

[50] A. Klapuri, "Analysis of musical instrument sounds by source-filter-decay model," in *Proceedings of the ICASSP*, 2007, pp. 53–56.

[51] B. Su, *An intergrated musical auto-transcription system*, Master Thesis, National Taiwan University, 2001.

[52] M. J. Kartomi, *On Concepts and Classifications of Musical Instruments*, University Of Chicago Press, 1990.

[53] J. D. Deng, C. Simmermacher, and S. Cranefield, "A study on feature analysis for musical instrument classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 2, pp. 429–438, 2008.

[54] S. Aksoy and R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 563–582, 2001.

[55] S. Essid, G. Richard, and B. David, "Instrument recognition in polyphonic music," in *Proceedings of the ICASSP*, 2005, vol. 3, pp. 245–248.

[56] J. Eggink and G. J. Brown, "Instrument recognition in accompanied sonatas and concertos," in *Proceedings of the ICASSP*, 2004, vol. 4, pp. 217–220.

[57] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Instrument identification in polyphonic music: Feature weighting with mixed sounds, pitch-dependent timbre modeling, and use of musical context," in *Proceedings of the ISMIR*, 2005, pp. 558–563.

[58] S. Essid, G. Richard, and B. David, "Instrument recognition in polyphonic music based on automatic taxonomies," *IEEE Transactions on Audio, Speech, Language Processing*, vol. 14, no. 1, pp. 68–80, Jan. 2006.

[59] J. Eggink and G. J. Brown, "A missing feature approach to instrument identification in polyphonic music," in *Proceedings of the ICASSP*, 2003, vol. 5, pp. 553–556.

[60] S. F. Chang, T. Sikora, and A. Purl, "Overview of the MPEG-7 standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 688–695, 2001.

[61] D. P. W. Ellis, C. V. Cotton, and M. I. Mandel, "Cross-correlation of beat-synchronous representations for music similarity," in *Proceedings of the ICASSP*, 2008, pp. 57–60.

[62] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[63] W. Pedrycz and F. Gomide, *Fuzzy Systems Engineering: Toward Human-Centric Computing*, Wiley-IEEE Press, 2007.

[64] C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines," 2001, [online] available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[65] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Instrogram: A new musical instrument recognition technique without using onset detection nor f0 estimation," in *Proceedings of the ICASSP*, 2006, pp. 14–19.

[66] H. T. Cheng, Y. H. Yang, Y. C. Lin, I. B. Liao, and H. H. Chen, "Automatic chord recognition for music classification and retrieval," in *Proceedings of the ICME*, 2008, pp. 1505–1508.