A Study on Music Similarity Analysis and Repeating Pattern Detection Based on Audio Fingerprinting

Chen Mei

A Thesis submitted to Tokushima University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

March, 2016

Department of Information Science and Intelligent Systems
Graduate School of Advanced Technology and Science
Tokushima University

Acknowledgment

Many people have offered me valuable help in my thesis writing, including my supervisor, my classmates, my parents, and so on.

First and foremost, I would like to show my sincere gratitude to my supervisor, Professor Kenji Kita, who was a responsible and resourceful scholar, gave me valuable guidance in every stage of the writing of this thesis by providing me with necessary materials, advice of great value, inspiration of new ideas, etc. It is his suggestions that draw my attention to a number of deficiencies and make many things clearer. Without his help, I could not have completed my thesis well. His keen and vigorous academic observation enlightens me not only in this thesis but also in my future study. Meanwhile, I would thank Professor Fuji Ren and Professor Masami Shishibori, who had contributed a lot of time and efforts in reviewing my thesis. And their valuable recommendations helped to improve this thesis.

Then, I would also like to thank all my teachers who have helped me to develop the fundamental and essential academic competence. Specially, I would thank Instructor Kazuyuki Matsumoto and Vice Professor Minoru Yoshida, who had contributed so much time in my doctoral terms. Besides, I would thank the Department of Information Science, Intelligent Systems, and etc. all teachers, who helped me a lot when I was taking my doctoral courses.

Besides, I am pleased to acknowledge the members of A2 group for their support, encouragement throughout the preparation of the original manuscript. Their suggestion on works and papers gave me much inspiration in my study.

Finally, I thank my family members who have given me their constant encouragement and support in my academic terms. Without all these people, I could not have completed the.

Contents

1 Introduction **1**

1.1 Background 1

1.1.1 Metadata-Based Music Retrieval 1

1.1.2 Content-Based Music Retrieval 2

1.1.3 Detection of Repeating Pattern 4

1.2 Motivation and Content of Research 5

1.3 Thesis Organization 6

2 Related Work **8**

2.1 Music Feature 8

2.2 Algorithms of Detecting Repeating Pattern 9

2.2.1 Symbolization Technique 10

2.2.2 Numeral Technique 11

3 Extraction of Feature **13**

3.1 Extracting Sub-fingerprint Feature 14

3.2 Related Concepts 16

4 A Fast Music Retrieval **19**

4.1 Related Works 19

4.1.1 Philips' Fingerprint Retrieval 20

4.1.2 Fibonacci Hashing Function 20

4.2 Proposed Method 25
4.2.1 Database Model 25
4.2.2 Search Model 27
4.3 Experiments and Analyses 28
4.3.1 Evaluation on Algorithmn 28
4.3.2 Analysis on Complexity 29
4.4 Summary 31

5 Similarity Analysis 37
5.1 Subsequent Similarities 38
5.2 Non-Similar Similarities 40
5.3 Similar Similarities 42

6 Detecting Repeating Patterns Based on Similarity Analysis 45
6.1 Detection Algorithm 45
6.1.1 Capturing Similar Blocks 46
6.1.2 Mergence 46
6.1.3 Boundary Refinement 47
6.2 Experiments and Analyses 48
6.2.1 Evaluation on Algorithm 48
6.2.2 Analysis on Performance 50
6.2.3 Analysis on Ability of Recognition 51
6.2.4 Analysis on Structure 54
6.2.5 Analysis on Complexity 55
6.2.6 Analysis on Redundancy Data 58
6.3 Summary 59

7 Contribution and Futher Works 60
7.1 Conclusion 60
7.2 Futher Researches 61

List of Figures

3.1 Extracting Sub-Fingerprint Feature 14

4.1 Philips' Database Model 21
4.2 Philips' Retrieval Model 22
4.3 Structure of Hash Table 23
4.4 Database Model 32
4.5 Retrieval Model 33
4.6 Average Search Time 34
4.7 Usage Rate 35
4.8 Average Usage Rate 36

5.1 Distribution of Subsequent Similarities 39
5.2 Distribution of Random Similarities 40
5.3 Distribution of Non-Similar Similarities 41
5.4 Distribution of Similar Similarities 43

6.1 Hash Table of Storing Similar Blocks 46
6.2 Element-wise Similarities in Repeating Patterns 51
6.3 Element-wise Similarities in A Similar Block Pair 52
6.4 Duration of Repeating Pattern 53
6.5 Overall Similarity of Repeating Pattern 54
6.6 Discovery Rate of Repeating Pattern 55
6.7 Similarity of Repeating Pattern 56

6.8 Distribution of Misalignment Error 57

6.9 Distribution of Redundancy Data 58

List of Tables

4.1 Values of C for Various Word Sizes 24

5.1 Domain of Non-Similar Similarities 42

6.1 Evaluation on Algorithm 49

Abstract

As the techniques of audio signal process are fast developing, the music resources are rapidly rising. Correspondingly, the techniques of music retrieval are changing from metadata-based music retrieval to content-based music retrieval.

The content-based music retrieval, as a high-level application compared with metadata-based music retrieval, can be applied in many of new scenarios. Therefore, it is widely employed and a lot of relevant algorithms are proposed. Thereinto, detecting repeating patterns is one of the most important applications in content-based music retrieval.

In this thesis, based on the sub-fingerprint feature, we study content-based music retrieval and lay stress on studying repeating patterns.

Currently, a lot of algorithms are presented to detect repeating patterns. However, current works mainly focus on the feature extraction, accuracy, the complexity of time and space, etc., while the similarities of repeating patterns are not deeply studied. As a matter of fact, a repeating pattern could be easily comprehended and distinguished by human beings. This represents that both repeating patterns and non-repeating patterns should have an approximate difference in similarities. This motivates us to explore a new solution for detecting repeating patterns from the aspect of similarity analysis.

In our method, we first segment a sub-fingerprint sequence of a piece of music recording into fixed-length blocks. Next, we study the distribution of similarities based on fingerprint blocks from three aspects: subsequent, similar and non-similar. Finally, according to results of similarity analysis, a related method is also proposed to detect repeating patterns. To evaluate the whole algorithm, experiments based on a test corpus of 30 familiar songs are carried out. The results indicate that our approach is promising.

Chapter 1

Introduction

1.1 Background

Music as one of the main cultures performs an important part in our daily lives and largely influences our daily lives. Especially, in the past decades, with the rapid development of the Internet, music information has become one of important information resources in the Internet. The music recordings are changing from previously fast increasing to now steeply rising. Accordingly, the dramatic shift also causes a technology diversion of music retrieval from metadata-based music retrieval to content-based music retrieval [1–5].

1.1.1 Metadata-Based Music Retrieval

In the early of music retrieval, people mainly make use of music metadata [1] [2] to retrieve music. In music retrieval, metadata usually refers to some important properties of the music object, such as title of music, artist, album, music type, place and date of publication, duration of music, and so on as basic elements to express a music object as accurately as possible. To achieve the effective search of music recordings, it is required that metadata vocabulary of describing music objects follows the same rules as far as possible, such as accurately, unambiguously,

easy-to-understand, succinctly and so on. To some extent, using the same rules can ensure that metadata descriptions are consistent in different managers or application scenarios. In addition, to achieve the generality of descriptions, such criteria as articulation, spelling, standard characters, etc., are also essential. Because metadata has the ability of rich expressions, it can basically meet most of requirements for many of general users. Until the present, metadata-based music retrieval is still the main method in many of music web sites or search tools, such as Google, Yahoo, Baidu Music, Apple iTunes, Amazon, and so on.

Because metadata mainly relies on manual annotations, the error easily occurs in describing music objects. Especially as the amount of music recordings is steeply rising, it becomes more difficult to manage metadata descriptions. Firstly, for different managers, they are accustomed to label music objects with individual conventions, so it easily leads to non-uniform descriptions. The non-uniform descriptions will seriously influence the search performance. Secondly, as music resources are gradually opening in Internet, for any user, they can not only freely download music resources, but also freely upload their favorite music resources to share with families, friends, relatives, and so on. In these scenarios, it is more difficult to supervise the metadata. Finally, with the increasing requirements, such as retrieving similar songs, humming, searching the unknown songs, accurately classifying, etc., it is becoming more and more difficult.

1.1.2 Content-Based Music Retrieval

To solve issues of metadata-based music retrieval, the content-based music retrieval [6–11] is proposed. In this method, the feature can be automatically extracted from the raw music signal by techniques of audio signal processing to represent the music object. Based on the feature sequence, it can achieve many of applications [12–29] , such as retrieving, classifying, identifying, managing, and so on.

In content-based music retrieval, the feature sequence can be seen as a condensed

digital summary of a music object. For the used features, it is not only required to accurately reflect the concerned information, but also is seen as a strict notion of similarity, which can be measured, computed, compared and so on. Currently, there are many classification methods about music features [6] [7]. In this thesis, we here introduce two categories: low-level features and high-level (sometimes also called semantic features).

The low-level feature mainly relates to acoustic properties of music signal, such as frequency, loudness, brightness, bandwidth and harmonicity. These features can be easily extracted by using the techniques of signal processing from music recordings.

The high-level feature [24] refers to melody, theme, motif, and so on. The high-level feature can allow more user-friendly descriptions and reduce the gap between low-features and semantic descriptions. In content-based retrieval, generally using a relatively small music clip is to achieve some relevant applications. In the content-based music retrieval, because the music feature is extracted by techniques of audio signal processing, the feature sequence of a music object is commonly considered as fingerprint of this music object [25–29]. Currently, there are many of important applications about the content-based music retrieval as follows.

Audio Identification

In content-based retrieval, audio identification is one of important applications. For example, using a short music clip within a piece of song is to identify a corresponding music recording. In the past decades, there are a lot of applications based on audio identification to be proposed, such as broadcast monitoring, music copyright protection, automatic management of music collections, tune identification, and so on.

Version Identification

Version identification is seen as music retrieval based on document-level, which is comparing similarities between entire documents. In real applications, an original recording may exist many of different versions, possibly having some changes in

timbre, harmony, melody, tonality, and so on. The aim of version identification is to identify different versions of the same music recording.

Audio Searching

Audio searching is one of the main applications in content-based music retrieval. In many cases, users may forget or do not know metadata of music, such as the tile of songs, singer, date, and so on. But, they may usually remember or know a tune, a melody, etc. At this time, users can hum a tune as a short query and send to a server for searching the corresponding song in a music database. Besides, there are also other important applications, such as category-based retrieval, structure analyzing of music, suspicious sounds, and so on.

1.1.3 Detection of Repeating Pattern

In content-based audio retrieval, discovering repeating patterns is one of important application fields [30–50] and becoming more and more popular. It is well known that most of songs have repetitive structures, so a study on repeating patterns is of important value in further studying and analyzing music. The main tasks of detecting repeating patterns are to find repetitive structures in the same song and that is also a prominent character compared with other applications of content-based audio retrieval. As a matter of fact, the repetition refers to difference in similarity based on some level of abstraction.

According to types of different features, the repeating pattern [30] is respectively defined as a set of ontime-timbre, ontime-chords, or ontime-pitch, etc. which occur at least twice with reference to the similarity measure. Because lengths of repeating patterns are usually much shorter than that of the music objects, repeating patterns can achieve some efficient applications in content-based music retrieval. Besides, repeating patterns are easily comprehended from perceptually similar and commonly considered as one of the most expressive and representative parts in music objects, so it is also seen as fingerprint of a music object in content-based music retrieval.

In a word, the repeating pattern is very helpful for further studying, analyzing and understanding music, such as music summarization [42], discovering themes [37] [38] and motifs [46–48], structure analysis [40], [45] [50] , music thumbnail [39, 40] and so on.

1.2 Motivation and Content of Research

In this thesis, we study content-based music retrieval based on the sub-fingerprint feature and mainly focus on repeating patterns by analyzing similarity. The whole tasks of our researches mainly contain two parts as the following.

(1) The first part

In the first part, the main tasks are studying the content-based music retrieval based on the sub-fingerprint feature and propose a fast music retrieval algorithm as an extension of Philips' method [29]. In this method, to search songs, a lookup table is exploited containing all possible sub-fingerprints as entries, but in fact it is unachieved in limited memory space. Beside, because the distribution of hash values is non-uniform in hash table and a lot of memory space is out of use. In our method, we first exploit Fibonacci Hashing function to provide a good distribution of hash values. Secondly, we operate the right shift to adjust the size of hash table according to the capacity of practical available memory. Finally, experiments are carried out to evaluate the proposed approach.

(2) The second part

In the second part, our aims are to study similarities of repeating patterns. Since repeating patterns are generally easily comprehended by human beings, so the study on repeating patterns is to help us further understand and analyze music.

Currently, many algorithms [30–50] were proposed to find repeating patterns. For example, L. Lu et. al. [33] mainly exploited a self-similarity matrix to analyze the music data and extract similar melodies. J. J. Aucouturier and M. Sandler [39]

used the techniques of image processing to look for repeating patterns. C. Wei and B. Vercoe [40] detected the repeating segments with fixed length and used heuristic rules to infer the structure. J. Lin et. al. [49] introduced a repeating discovery algorithm based on the discrete representation of non-trivial motifs in the time series. J. Paulus and A. Klapuri [50] presented a structural description method to detect the repeating parts in the music. However, previous works are mainly focusing on how to discover repeating patterns, improve the complexity of time and space, and etc. but there are few literatures about studying similarities of repeating patterns. This thesis focuses on similarities of repeating patterns. Since lengths of repeating patterns considerably differ, it is very difficult to directly analyze repeating patterns. In [29], a basic unit, which is containing 256 subsequent sub-fingerprints corresponding to a granularity of 3 seconds, is used to identify a song. In this thesis, we also consider that a pair of 256 subsequent sub-fingerprints can identify a corresponding block within a repeating pattern. In reality, the similar block pairs can be seen as shorter repeating patterns. This also gives a new idea, namely, segmenting repeating patterns into blocks to study. Therefore, we firstly segment a fingerprint sequence into blocks with fixed-length, then analyzes similarities of block pairs, and finally presents a related approach to find repeating pattern according to these analyses.

1.3 Thesis Organization

This thesis mainly studies content-based music retrieval and puts stress on detecting repeating patterns. The work of the whole thesis includes two parts and is organized in the rest chapters as follows.

Chapter 2: Related Work

In this chapter, we introduce some works and techniques relevant to our researches from two aspects: extraction of music features and techniques of music

retrieval, mainly focusing on detecting repeating patterns.

Chapter 3: Extraction of Feature

In this chapter, the goal is to introduce extraction of music features. In our method, the sub-fingerprint feature based on the method of Philips' method is exploited. So, in this chapter, we simply review their method and introduce some relevant concepts.

Chapter 4: A Fast Music Retrieval

In this chapter, based on the sub-fingerprint feature, a fast retrieval algorithm is proposed to search songs as an extension of Philips' method. In our method, Fibonacci Hashing function is exploited to provide a good distribution of hash values and the size of hash table can be adjusted according to the practical memory space.

Chapter 5: Similarity Analysis

In this chapter, we analyze the distribution of similarities based on fingerprint blocks with fixed length from three aspects: adjacent relationship, non-similar relationship, and similar relationship, by experiments.

Chapter 6: Detecting Repeating Patterns Based on Similarity Analysis

In this chapter, according to similarity analysis of Chapter 5, we present a new method to detect repeating patterns. Firstly, based on similarity analyses of Chapter 5, a relevant method is used to capture the similar blocks and these similar blocks are stored into a hash table. Next we further refine these similar blocks to extract repeating patterns. Finally, we evaluate the proposed method by experiments and fully analyze the experimental results.

Chapter 7: Contribution and Future Work

In this chapter, we conclude the works of this thesis and make a prospect our future work.

Chapter 2

Related Work

In this thesis, our researches include two parties. The first part introduces a fast retrieval music based on Philips' method [29] and reviews Philips' method in Chapter 4. In the second part, we mainly study repeating patterns by analyzing similarity and this is the main emphasis of our research. So in this chapter, we mainly describe some works relevant to repeating patterns.

In detecting repeating patterns, the whole algorithm usually contains two phases: the goal of the first phase is extracting features; in the second phase, according to the used features, a relevant algorithm is exploited to find repeating patterns. In this chapter, based on two phases, we introduce some relevant works.

2.1 Music Feature

In feature extraction, the selection of feature is a crucial step and directly relates to the techniques of music searching. At present, there are a variety of classification methods about music features [6] [7] [24] , such as acoustical features, the basic features, derived features, thematic features, etc.

Acoustical features are mainly related to the human auditory perception, such as loudness, pitch, bandwidth, harmonicity, brightness, etc., which can be easily derived from the raw music recordings by techniques of signal processing. Loudness

feature can be approximately computed through the square root of the energy of the signal. Pitch feature, which is measured based on short-time Fourier Spectra, is obtained from frequency and amplitudes in the peaks. Brightness can represent the higher frequency of the music signal and is calculated by the centroid of the magnitude spectra of the short-time Fourier. Bandwidth is a measure of the width of frequencies and mainly related to between the spectral components and the centroid of the short-time Fourier transform. Harmonicity represents the degree of acoustic periodicity, can be used to distinguish harmonic spectra, inharmonic spectra, and noise, and is measured by the deviation of the sound spectrum.

For the basic features [29], they can easily be extracted from music objects, such as Mel-Frequency Cepstral Coefficient, Frequency Cepstral Coefficients, Fourier Coefficients, Spectral Flatness, Sharpness, etc. But, in practical application, these basic features are usually further processed to derive some of new features.

The derivation features, which are derived from basic features, are widely used, such as Derivatives, Means, Variances, CQT [45], Sub-fingerprint [29], etc.

Thematic features [17] include themes, melodies, motifs, and so on. For example, Chih-Chin Liu et. al. [38] used the longest repeating as theme within a music recording, which is based on a sequence of note features. These works [45] [46] extracted perceptual attributes to represent melody features. The melody features, and the chord features can be easily derived from the basic features.

2.2 Algorithms of Detecting Repeating Pattern

After selecting features, extraction of feature is performed by using techniques of music signal processing and a corresponding feature sequence is generated. According to types of the features, some of relevant algorithms can be used to detect repeating patterns. The music signal can be generally converted into two representations: symbolization representation and numeral representation. Besides, these two

representations can also convert each other by quantification. Therefore, techniques of discovering repeating patterns can be classified into two categories: symbolization techniques and numeral techniques.

2.2.1 Symbolization Technique

In symbol representations, audio signal is converted into symbols by a series of techniques of signal processing. Of course, some numeral representations can be also transformed into symbol representations by quantification. After generating the symbol feature sequence, then these techniques for processing symbols can be applied, such as string match, tree, suffix tree [31], K-means [32], clustering algorithms [33], etc.

For example, in [34], Ning-Han Liu1 et al. chose the information of pitch and duration as features based on the MIDI to form a sequence of triples in the on-time note and a modified R*-tree was used to filter impossible candidate ARP. In literature [35], Jia-Lien Hsu et al. used notes of a music object to form symbol features, constructed a correlative matrix to store the intermediate results, and finally extracted repeating patterns from the intermediate results by a string-join operation. In work [36], Ioannis Karydis et al. used the note sequences as feature, exploited an aggressive accession to find all intermediate patterns with maximum-length, and finally refined these intermediate patterns to extract all true repeating patterns. Chaokun Wang et al. [37] used character features based on a sequence of notes and proposed an index structure called N-gram to mine theme in a piece of music.

Chih-Chin Liu et al. [38] extracted three features: rhythm, melody, and chords as audio feature, and constructed a data structure called 1D-List to perform the similar string matching. Jean-Julien Aucouturier and Mark Sandle [39] extracted features of timbres, segment the feature sequence into a meaningful succession of blocks and used two techniques of Kernel Convolution and Hough Transform to

detect repeating patterns.

2.2.2 Numeral Technique

In numeral representations, music signal is transformed into numeral representation. Besides, some of numeral representations can be also converted into symbol representations. Numeral representations include univariate time series, multivariate time series and so on. For numeral techniques, it is based on the concept of strict similarity and can be measured, computed and so on.

To find repeating patterns in numeral representations, a lot of algorithms are also proposed. For example, in work [13], Guodong Guo and Stan Z. Li used perceptual features, which were consisted of total power, brightness, MFCCs, etc., and the SVMs was proposed to detect repeating recognition. Wei Chai and Barry Vercoe [40] extracted time vector series, detected the fixed-length repetitions, and used heuristic rules to infer the repetitive structure. Masataka Goto [41] extracted chroma features to represent chorus of music and proposed a method called RefraiD to repeating patterns.

Besides, in algorithms of discovering repeating patterns, the self-similarity matrix or vector was widely used [42–48]. For instance, in work [42], a 2-D similarity matrix was used to find given-length repetitions. In [43], Bartsch extracted chorus to form a new feature set based on quantized chromagram and used a self-similarity matrix to extract repeating patterns. Foote [44] was based on a note sequence, such as verse and chorus and built a self-similarity matrix to locate points of obvious change (namely looking for similar elements), and exploited the techniques of image processing to extract the repeating musical patterns. Lie Lu et al. [45] used CQT as features, constructed a self-similarity matrix to capture the similar elements and extracted all significant repeating patterns based on the structure analysis of acoustic music data and techniques of image processing.

However the self-similarity vector is applied widely, this method requires a com-

plexity of $O(2^n)$ for memory and computational requirement. To reduce computational cost, Lei Wang [46] proposed an algorithm called Adaptive Motif Generation to capture the candidate motifs and constructed a sparse self-similarity matrix to further refine these candidate motifs for extracting the final repeating patterns.

Chapter 3

Extraction of Feature

The aim of this chapter is to introduce extraction of music features. Before extracting features, we first need to select appropriate features relevant to our research. In fact, selection of features usually depends on actual and real applications, such as focusing on timbral, tempo, beat, loudness, pitch, brightness, rhythm, theme, motifs, melody, chord, etc.

In this thesis, we will mainly focus on melody information, so the used features must be able to accurately express melody mainly relevant to perceptual features. Firstly, the feature sequence can be considered as a perceptual digest of the music recording, retaining the relevant information of a real music object as far as possible. Secondly, it should be more invariant to kinds of distortions, such as background noise, D/A-A/D conversion, audio coders (such as GSM and MP3), compression, signal degradations, noise addition, and so on. Thirdly, given an enough long subsequence, it should identify a corresponding song. Finally, the operations on feature data are easily computable.

In these literatures [45, 46] , they also focused on melody, CQT and chroma features were used, and experimental results have showed that the repeating patterns can be discovered well. In literature [29], Jaap Haitsma and Ton Kalker proposed a sub-fingerprint feature and now it is widely used. So this paper uses this feature.

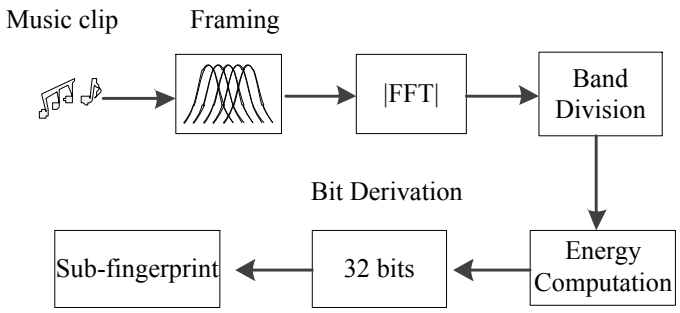

Figure 3.1: Overview of Haitsma and Kalker's Audio Fingerprint.

The scheme of feature extraction is reviewed in the following subsection.

3.1 Extracting Sub-fingerprint Feature

An overview of Haitsma and Kalker's audio fingerprint extraction algorithm is as shown in Figure 3.1.

Figure 3.1 shows the whole process of sub-fingerprint extraction. Here we only take into the main phases of the whole process.

(1) Preprocessing

The first phase is preprocessing. The audio signal is converted to a general format: mono PCM (16 bits), a fixed sampling rate of 44.1 KHz, and MP3 formats.

(2) Framing

The second phase is framing. The goal of this phase is to divide the whole audio signal in overlapping frames. In framing, a 31/32 overlap factor is used to allow 5.8 milliseconds off with respect to the real frame boundaries and assure that even in the worst cases, both the query sub-fingerprints and the sub-fingerprints of the same clip in database are still very similar. Due to using the large overlap, subsequent sub-fingerprints are very similar and slowly reducing in time. Besides, each frame is weighted by a Hanning window.

(3) Extracting frequency sub-bands

The third phase is performing the Fourier Transform on every frame. Because the frequency domain contains usually the most important perceptual features of music objects, a time-domain representation of raw audio data is converted into a spectral representation by carrying out a Fourier transform. In fact, the frequency domain from 300Hz to 2000Hz is the most related frequency range of human hearing. Therefore, the frequency domain for each frame is divided into 33 non-overlapping sub-bands from 300Hz to 2000Hz by using a logarithmic spacing.

(4) Deriving bit

The fourth phase is deriving bit. The derivation is based on the sign of energy differences between subsequent frequency sub-bands. Let $E(n, m)$ denote the energy of frequency band m of frame n and $f_n(m)$ denote the m-th bit of the sub-fingerprint of frame n, then $f_n(m)$ is formally defined as Eq.3.1:

$$f_n(m) = \begin{cases} 1, if E > 0 \\ 2, if E \leqslant 0 \end{cases}. \tag{3.1}$$

Where

$$E(n, m) = E(n, m) - E(n, m + 1) - E(n - 1, m) + E(n - 1, m + 1). \tag{3.2}$$

(5) Generating the sub-fingerprint

The fifth phase is generating the sub-fingerprint feature. In the fourth phase, 32 bits are obtained by calculating 33 subsequent sub-bands for each frame and then form a sub-fingerprint. Finally all sub-fingerprints of a music signal form a feature sequence.

Because each bit can reflect the energy differences of two subsequent frequency sub-bands, the sub-fingerprint feature can reflect change of melody. Secondly, this method has been demonstrated more robust against various "corrupted" inputs

such as compressed, delayed music, etc. Besides, the sub-fingerprint representation with 32-bit is more compact compared to raw multidimensional features, while the calculation is much simpler. Therefore, the sub-fingerprint feature based on Haitsma and Kalker's is more suitable in our application.

Let a sub-fingerprint sequence $FP = f_1 f_2 ... f_N$ and it is consisted of N sub-fingerprints. The duration of each sub-fingerprint is equal (about 0.01s), so the product both sub-fingerprint index and rate of sampling can approximately represent timestamp.

3.2 Related Concepts

In practical applications, however, a single sub-fingerprint is usually too short to contain sufficient information. To obtain sufficient information, a fingerprint block as a basic unit, which is also a subsequence of the entire feature sequence, is commonly used to identify one song.

Let $FP = f_1 f_2 ... f_N$ represent a sub-fingerprint sequence, then the i-th fingerprint block FP_i^L is described as Eq.3.4:

$$FP_i^L = f_i ... f_{i+1-L}, i \in [1, N - L]. \tag{3.3}$$

Where L denotes the number of sub-fingerprints in FP_i^L (also called block length); N denotes the number of sub-fingerprints in FP; f_i denotes the i-th sub-fingerprint of FP. Moreover, bit error rate is used to express the robustness, and Hamming distance is used to calculate the distance or similarity between fingerprint blocks. Let FP_i^L and FP_j^L represent two fingerprint blocks, their distance is described as Eq.3.4:

$$BER(FP_i^L, FP_j^L) = \frac{\sum_{l=0}^{L-1} \sum_{m=0}^{31} f_{i+l}(m) \wedge f_{j+l}(m)}{32 \times L}. \tag{3.4}$$

Where m denotes the m-th bit of the sub-fingerprint; and $\wedge$ is bit operator XOR (exclusive OR). It is noted that the smaller the BER of two fingerprint blocks, the higher the similarity of the corresponding fingerprint blocks is. The domain of BER is from 0 to 1.

In our thesis, the interval distance between fingerprint blocks short for ID is often used. Here we describe this concept. For two fingerprint blocks FP_i^L and $FP_j^L(i < j)$, then $ID(i, j)$ is defined as Eq.3.5:

$$ID(i, j) = j - i. \tag{3.5}$$

Besides, there are also other definitions relevant to our research as the following descriptions. Here let FP_i^L and $FP_j^L(i < j)$ represent two fingerprint blocks.

Definition 1

Similar threshold: If meeting the condition $BER(FP_i^L, FP_j^L) \leq \rho$, this equation can present similar relationship between fingerprint blocks, and then the value is called the similar threshold.

Definition 2

Similar block: If meeting $BER(FP_i^L, FP_j^L) \leq \rho$, then FP_i^L and FP_j^L represent a pair of similar blocks, then FP_i^L and FP_j^L are called similar blocks each other. As a matter of fact, a pair of similar blocks is also regarded as a shorter repeating pattern.

Definition 3

The longest repeating pattern: The longest repeating pattern is considered as being composed of many of subsequent fingerprint blocks.

Definition 4

Trivial repeating pattern: In the longest repeating patterns, if a pattern is a sub-pattern of another pattern, then the sub-pattern is called trivial repeating pattern.

Definition 5

Non-trivial repeating pattern: In the longest repeating patterns of a music object,

except for trivial repeating patterns, the left are called the non-trivial repeating patterns. In this thesis, if not giving special statements, repeating patterns refer to non-trivial repeating patterns.

Chapter 4

A Fast Music Retrieval

In this chapter, based on the sub-fingerprint feature, we first introduce a fast music retrieval algorithm to search songs as an extension of Philips' method [29]. On the one hand, Fibonacci Hashing function is employed to further process sub-fingerprints for providing a good distribution of hash values, which can effectively utilize memory. On the other hand, the right shift operation is performed to adjust size of the lookup table. In conclusion, the performance of presented algorithm is evaluated by experiments. The process is described in detail in the following subsections.

4.1 Related Works

As above-introduced, Jaap Haitsma and Ton Kalker proposed the sub-fingerprint feature and built a lookup table containing all possible sub-fingerprints as entries to search songs. In fact, if not considering the limited memory, for the sub-fingerprint feature, such a lookup table is the best ideal approach. But, in practical application, it is unachievable in limited memory space. In this chapter, we propose a method based on Jaap Haitsma and Ton Kalker's method to solve this defect. Therefore we firstly review the method of Jaap Haitsma and Ton Kalker in this section.

4.1.1 Philips' Fingerprint Retrieval

In work [29], based on the sub-fingerprint feature, a lookup table as index is exploited to retrieval songs. In order to contain all sub-fingerprints, it needs to allocate a memory containing 2^{32} sub-fingerprints. The layout of fingerprint database is shown in Fig.4.1 and the layout of search model is shown in Fig.4.2.

The whole process of Philips' fingerprint retrieval is illustrated in Figure 4.1 and Figure 4.2. In the lookup table, all possible sub-fingerprints with 32-bits are seen as entries, of which each entry is pointing to a list. In the list, each node contains information of pointers, in which the real sub-fingerprints are located. But, in practical application, the memory allocation of 2^{32} sub-fingerprints is unachievable. However hash table is used to take place of a lookup table by sparsely filling, in searching, these hash tables needs respectively to load into high-speed memory.

In fact, it is proved that hash table is a very effective and efficient method in a lot of application scenarios and widely used. In case we do not consider the restrictions of memory, for the sub-fingerprint feature, hash table should be an ideal method.

4.1.2 Fibonacci Hashing Function

This subsection will introduce an important technique called Fibonacci Hashing function, which belongs to Hash Table. Before describing, we first look back at the knowledge of Hash Table.

Hash table as an effective data structure [51] is widely employed to implement many of relevant applications. Practically, hash table is usually considered as a special array, which can map keys to values. If giving a key or input, by using hash function, it can generate a hash value and this hash value is usually as entry to achieve some relevant applications.

Figure 4.3 shows a structure of Hash Table. Ideally, in Hash Table, it should meet the two basic conditions as following:

(A) Each key is assigned to a unique hash value or bucket.

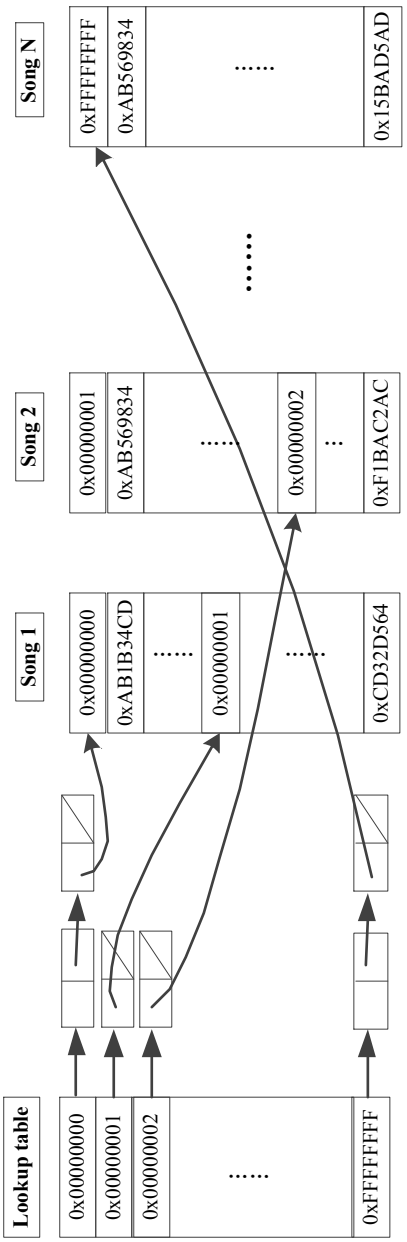

Figure 4.1: Philips' Database Model.

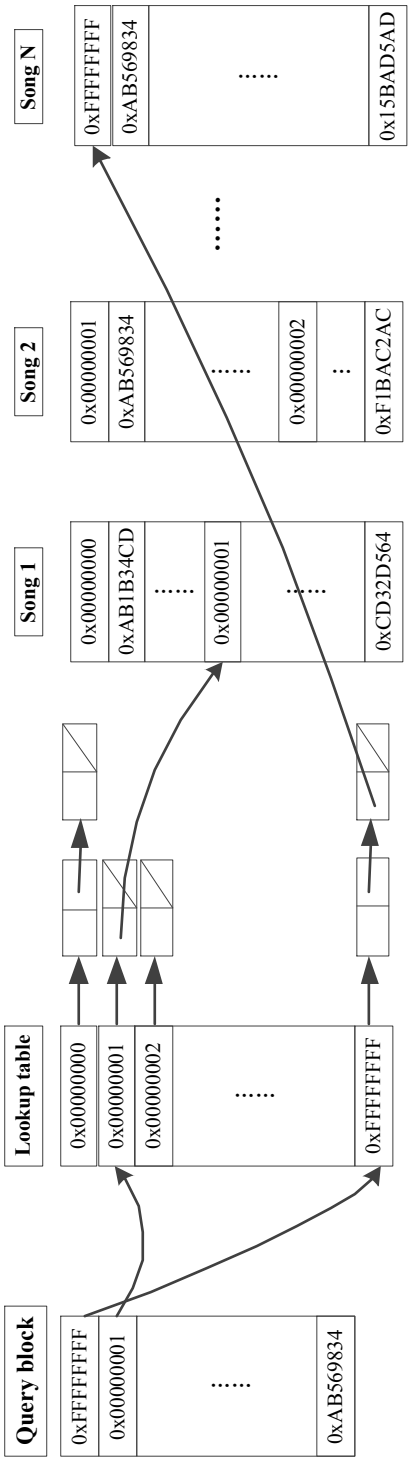

Figure 4.2: Philips' Retrieval Model.

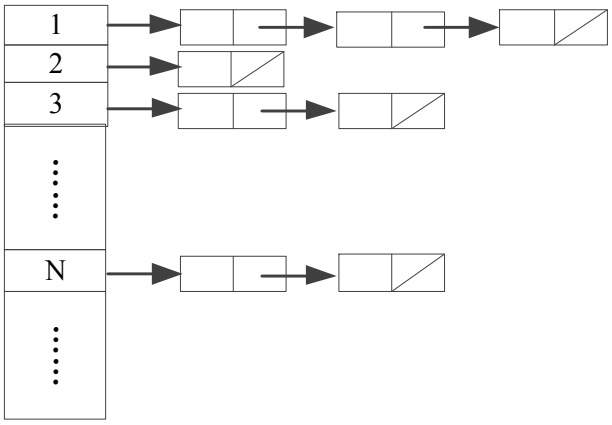

Figure 4.3: Structure of Hash Table.

(B) Hash values can evenly distribute across hash table.

But, in practical application, for these ideal cases, it is hardly achievable, in other words, hash collisions [52] would be inevitable. Hash collisions are generally described such that two different keys are assigned to the same entry. In order to solve hash collisions, some of related strategies have been proposed. In Hash Table, how to design a good hash function is a very crucial problem. On the one hand, it helps to address hash collisions. On the other hand, it can provide a uniform distribution of hash values [53] , which can further reduce the probability of collisions. Fibonacci Hashing function [54, 55] has been proved that it can generate a uniform distribution of hash values. In Fibonacci Hashing function, one important task is to find a very special value C [53] and multiply by key to generate the hash values. According to these descriptions, then Fibonacci Hashing function is defined as Eq.4.1:

$$f(key) = key * C. \tag{4.1}$$

A good distribution of hash values can be obtained by Eq.4.1. Where the value

W	C
2^{16}	40503
2^{32}	2654435769
2^{64}	11400714819323198485

Table 4.1: Values of C for Various Word Sizes.

of C is mainly related to the golden ratio [56, 57] . Given two positive numbers x and y, the golden ratio is described: the ratio of x to y is equal to that of $x + y$ to x. This ratio is called the golden rate. Let $\lambda = \frac{x}{y}$ represent the golden rate, then $\lambda = \frac{x}{y}$ is calculated as Eq.4.2:

$$\left.\begin{array}{r} \frac{x}{y} = \frac{x+y}{x} \\ \lambda = \frac{x}{y} \end{array}\right\} \Rightarrow \lambda = \frac{1+\sqrt{5}}{2}. \tag{4.2}$$

Where the value $\lambda = \frac{1+\sqrt{5}}{2}$ is just equal to the coefficient of Fibonacci number [58–60] . Thus the n-th Fibonacci number is given as Eq.4.3:

$$F_n = \frac{1}{\sqrt{5}}(\lambda^n - (\lambda^{-1})^n). \tag{4.3}$$

Where λ^{-1} is the reciprocal of λ. It has been proofed that the value of C is just closest to the part integer of $W\lambda^{-1}$ [53] and an idea special value. Where W denotes word sizes. Table 4.1 gives the values of C for various word sizes, with reference to literature [60]. Because the value of C relates to Fibonacci numbers, this hash table based on $W\lambda^{-1}$ is called Fibonacci Hashing Function.

Why do we choose $W\lambda^{-1}$ as the value of C. We here show the function of the value $W\lambda^{-1}$. As a matter of fact, the value $W\lambda^{-1}$ can divide the whole hash table into two parties by the golden ratio, for subsequent keys, they follow such a rule that each hash value falls into one of the largest remaining intervals. Furthermore, for each newly added hash value, it divides the interval to which it belongs according to the golden ratio. Additionally, because of using C in hash function, it can ensure that a good distribution of hash values is generated.

4.2 Proposed Method

In Philips' method, a lookup table containing all possible sub-fingerprints with 32-bits is exploited as the index. On the one hand, due to the limited memory, hash table instead of a lookup table will be usually sparsely filled. On the other hand, the distribution of hash values is non-uniform in hash table and a lot of memory space is out of use, which would cause a lot of waste. For example, for a hash table containing approximately 250 million sub-fingerprints, then the average usage rate is only 0.058 based on statistics of Philips' research. In fact, even uniform distribution, the rate is far less the size of 2^{32}. Therefore, in this chapter, we propose a method to solve the above-mentioned problems. Firstly, Fibonacci Hashing function is used to generate a good distribution of hash values in hash table. Secondly, the right shift operation is carried out to adjust the size of the lookup table according to the memory space.

4.2.1 Database Model

In subsection, we present a method based on Fibonacci Hashing function to build a model of music database. In this method, we use Eq.4.4 to process keys to generate hash values as the following:

$$f(key) = (key \times C) \gg N. \tag{4.4}$$

Where key refers to the sub-fingerprint and is the right shift operator. In our approach, the feature sequence is consisted of subsequent sub-fingerprints with 32 bits, of which each sub-fingerprint contains 4 bytes. According to Table 4.1, C is rightly equal to 2654435769. Besides, in Eq.4.4, N is the number of right shift bit and its domain is from 1 to 31. In application, the value of N mainly depends on the available memory space and the amount of songs, which can be flexibly adjusted. Through Eq.4.4, the value $FP * C$ is first moved N-bit to right. As a

result, sub-fingerprints of the same top $(32 - N)$-bit will be assigned to the same bucket. Thus size of hash table becomes 2^{32-N}. Besides, because the value C is used in hash function, it can ensure that a uniform distribution of hash values would be generated.

Through Eq.4.4, we can obtain a good distribution of hash values and flexibly modify size of hash table according to requirements. Next we introduce how to build this database model.

In practical application, the amount of songs usually achieves tens of thousands. If all sub-fingerprints are put into memory, it would be infeasible. To address this problem, our method is to design two hash tables, respectively called auxiliary hash table (AHT) and offset hash table (OHT). In AHT, it stores the hash values, which are generated by Eq.4.4. OHT stores offset information based on AHT, which can be obtained by computing the number of nodes in each entry. In each node, it is composed of the data domain and pointer. For the data domain, it contains three components: a real sub-fingerprint (FP), location of this sub-fingerprint in its fingerprint file and name of this fingerprint file (FN) of this song; in the pointer domain, it points to the next sibling node. While the data domain of OHT just contains offset. The database model is as shown in Figure 4.4 and the steps are as follows.

Step 1

By using Eq.4.4, AHT is first created.

Step 2

OHT is created based on OHT. Firstly, scanning AHT from top to bottom and from left to right, all list nodes are numbered. The label of the first node in each entry will be stored to the same entry in OHT.

Step 3

After establishing the database model, in auxiliary hash table, information of only data domain is written into a file called AHTF by the location number order.

After finishing, the auxiliary hash table is destroyed; in OHT, information of hash table is also written into a file called OHTF and OHT is destroyed.

4.2.2 Search Model

In the above subsection, we introduce the database model based on Fibonacci Hashing function. In this subsection, we introduce how to search songs exploiting this database model. In our method, the layout of search model is designed as shown in Figure 4.5. The operations of searching songs are illustrated as follows.

Step 1

Extracting the sub-fingerprint sequence from the query music clip as the query sub-fingerprints.

Step 2

Loading the file OHTF to create OHT.

Step 3

Selecting a sub-fingerprint from the query sub-fingerprints and exploiting Eq.4.4 to process generate a hash value.

Step 4

According to this hash value, it is locating corresponding entry and reading offset information in OHT.

Step 5

Based on offset information, the data of a specified rang in AHTF is loaded as the candidate sub-fingerprints, which are used to compare with this hash value.

Step 6

If having candidate sub-fingerprints is equal to the given sub-fingerprint, then it will continue comparing their corresponding sub-fingerprint blocks by calculating Hanning distance. If the distance is less than the threshold, they successfully match. Otherwise, it goes on comparing the next candidate sub-fingerprint until the end. After finishing, the next sub-fingerprint in the query sub-fingerprints is operated

using the same steps.

Step 7

After finishing the query sub-fingerprints, the top n songs with the minimum distance as query results is outputted.

4.3 Experiments and Analyses

In this chapter, we present a method to retrieve songs based on Philips' method. For one thing, we employ the Fibonacci Hashing function to generate a good distribution of hash values. For another, by the right shift, the memory space can be flexibly adjusted according to the available memory. In order to know about performance of the proposed algorithm, we evaluate from three aspects: search accuracy, search time and memory usage by experiments.

The experimental configuration is as described as follows.

Database

In experiments, we select a test corpus, which is containing 7605 music clips in mp3 format from the Internet. Besides, in our preliminary test, to reduce the other interference factors, we here choose some songs with low distortion. Genres of songs contain pop, classical, folk music, etc.

Hardware configuration

In the experiment, the total memory is 4G, with about a free memory of 1.6G.

System configuration

Ubuntu 11.04 version.

4.3.1 Evaluation on Algorithmn

In this subsection, we evaluate the performance of our approach by analyzing accuracy.

In our research, the sub-fingerprint feature with 32-bits based on Philips' method

is used. First, the raw audio signal is divided into overlapping frames with an overlap factor of 31/32. Every frame is weighted by a Hanning window. Next, it selects 33 non-overlapping frequency sub-bands from 300Hz to 2000Hz and generates 32 bits as a sub-fingerprint from every frame by comparing the energy difference between subsequent frequency sub-bands. Thus 32-bits form a sub-fingerprint, corresponding to about the interval of 11.6 milliseconds. Finally, a fingerprint sequence is generated based on these sub-fingerprints. Besides, to identify a song, a granularity of 256 subsequent sub-fingerprints as a basic unit is used in our method, corresponding to duration of only 3 seconds.

Next, by our improvement, the database model is created. Finally, we exploit the search model based on the experimental settings to analyze our method.

We extract about 61 million sub-fingerprints from pieces of 7605 music clips. In the experiments, we randomly select 100 short audio clips, perform the same operations 10 times, and finally compute the average accuracy. The final results show that the search accuracy can approximately achieve 100%. That means that our method is feasible.

In our method, because the right shift is used, for heavily degraded signals, it needs to consider more candidate sub-fingerprints. So we here only use the low degraded signals. For heavily degraded, it is our further focus.

4.3.2 Analysis on Complexity

In the above subsection, it has verified that our method can search songs well. Next we further analyze its efficiency of time and space.

Search Speed

In our method, by using the right shift operation, the size of hash table can be flexibly adjusted. Next we will observe the efficiency of time as the change of the value N is. Besides, in experiments, we only consider average search time. The experimental results are shown in Figure 4.6.

The change of average search time is as shown in Figure 4.6. The average search time is relatively stable about 0.97s. In reality, there are three reasons to explain. Firstly, the offset hash table is very small, can completely reside in memory and achieve high-speed retrieval. Secondly, in comparing, only specified data is loaded from AHTF. Finally, through Fibonacci Hashing function, the hash values can uniform distribute across the whole hash table. Therefore, there improvements can achieve fast searching.

Memory Usage

In this subsection, we analyze the memory usage. In order to observe the efficiency of memory usage, we here employ two standards to evaluate: usage rate (UR) and average usage rate (AUR).

Let UR represent the ratio of used entries in the total entries; UN denotes the number of used entries in hash table; L stands for the size of hash table. The calculation of UR is as follows:

$$UR = \frac{UN}{L}. \tag{4.5}$$

Let UR_N denote the proposed method, and UR_P for Philips' method. The experimental results are as shown in Figure 4.7.

UR_N is remarkably higher than UR_P as illustrated in Figure 4.7 and that can explain well that the distribution of hash values is relatively uniform in the whole hash table, namely Fibonacci Hashing function is a good hash function.

AUR denotes the rate of the number of sub-fingerprints and size of hash table. As mentioned-above, the amount of songs can also affect the choice of N. If songs are quite many and size of hash table is small, every entry will contain more nodes. As a result, times of comparison will increase. If songs are very few and size of hash table is large, memory space will be wasted a lot. Therefore, we here use the standard AUR to evaluate. Let AUR represent average usage rate; T stands for the total number of sub-fingerprints from database; L is size of hash table. Calculation

of AUR is as follows:

$$AUR = \frac{L}{T}. \tag{4.6}$$

The experimental results are shown in Figure 4.8 .

In Figure 4.8 , the AUR is slowly increasing from 5 to 9 and fast rising at $N = 9$. Obviously, the larger AUR is, the more the time of comparison is.

By comparing both Figure 4.7 and Figure 4.8 , in our experiments, the most appropriate value of N is 8.

4.4 Summary

In this chapter, we proposed a fast retrieval algorithm based on Fibonacci Hashing function as an extension of Philips' method. Firstly, we use Fibonacci Hashing function to generate a good distribution of hash values. Secondly, we exploit the right shift operations to flexibly adjust the size of hash table according to the practical memory space. The final experiments show that our method can search songs well. Besides, we also further analyze the efficiency of time and space and the results show that our improvements are also obvious.

But, because the right shift operation is used, for the high distortion, it leads to more candidate sub-fingerprints compared with Philips' method. So this is drawback of our approach. In the future, we focus on how to solve this drawback.

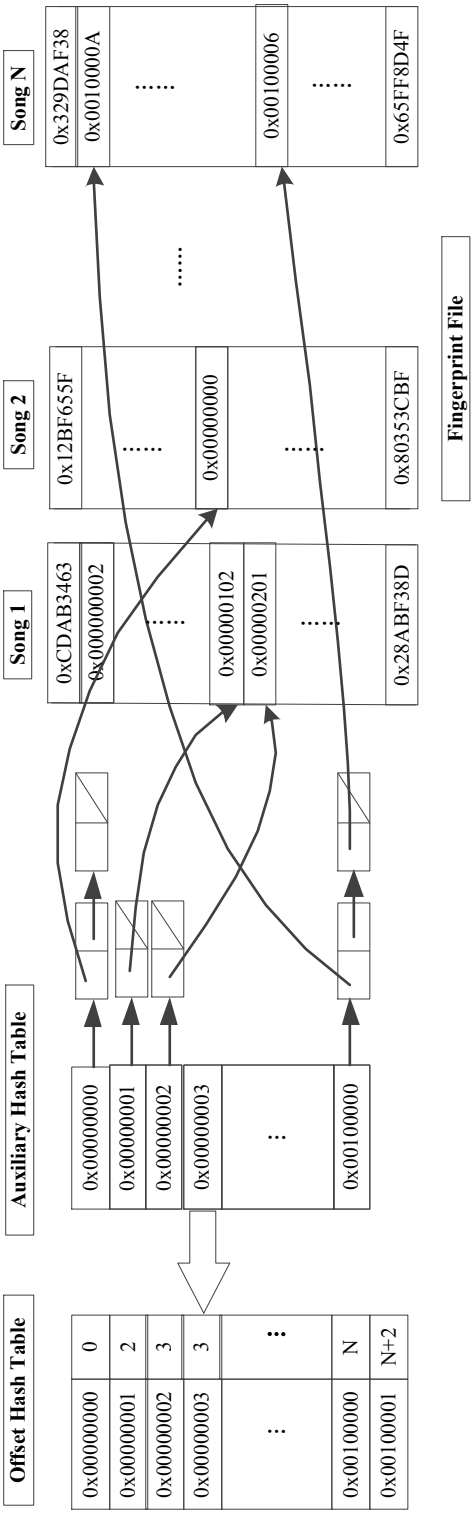

Figure 4.4: Database Model.

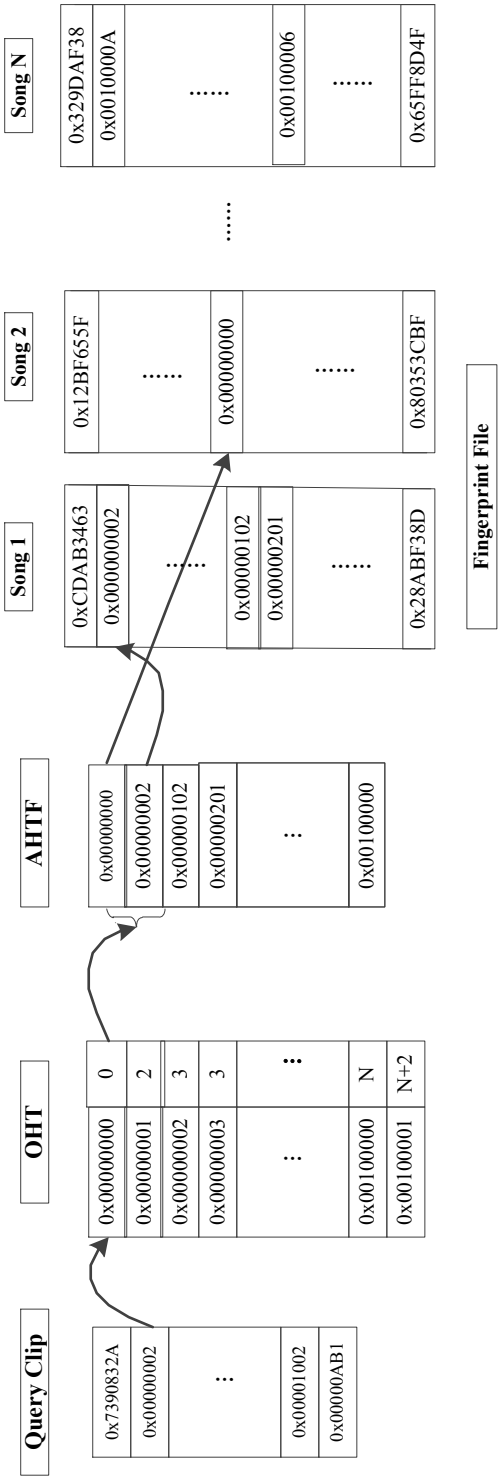

Figure 4.5: Retrieval Model.

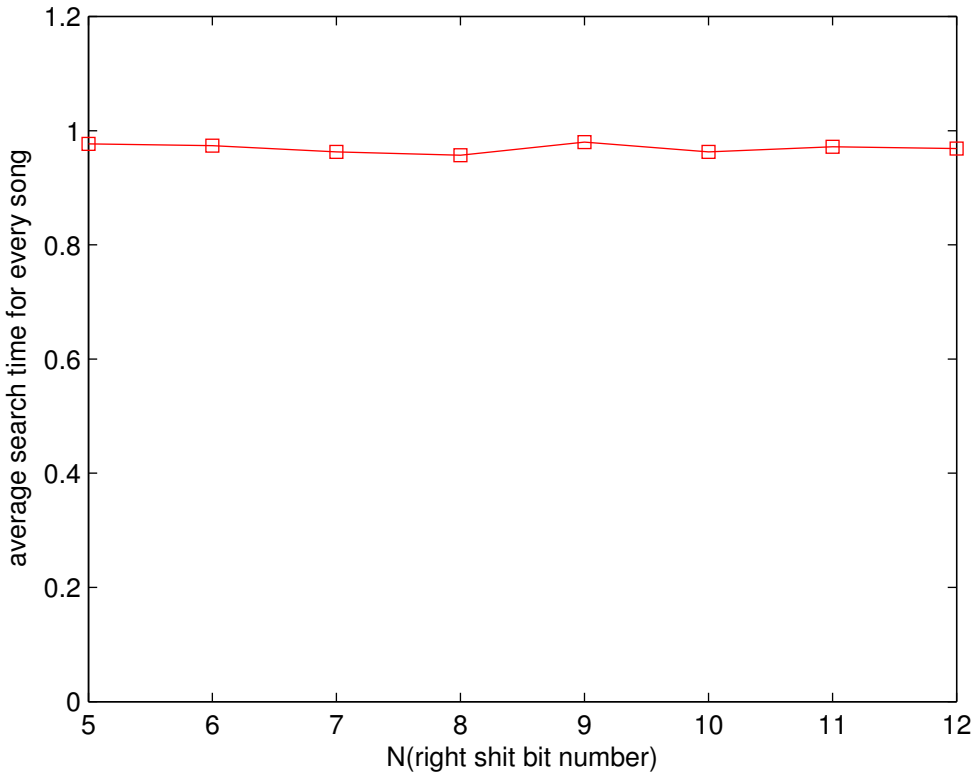

Figure 4.6: Average Search Time.

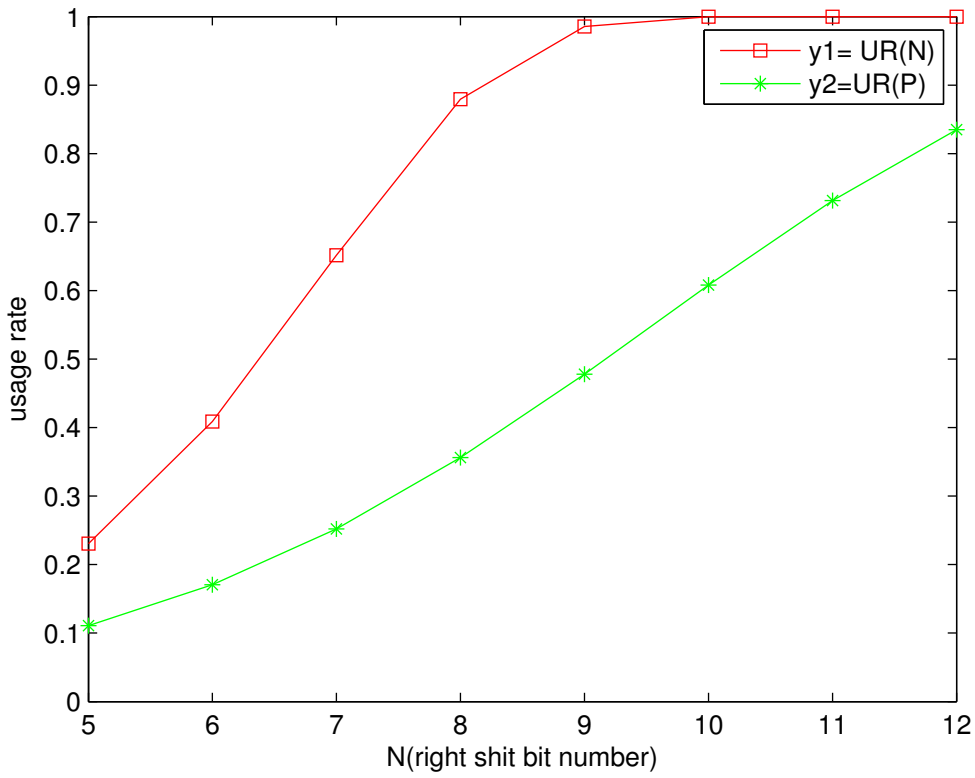

Figure 4.7: Usage Rate.

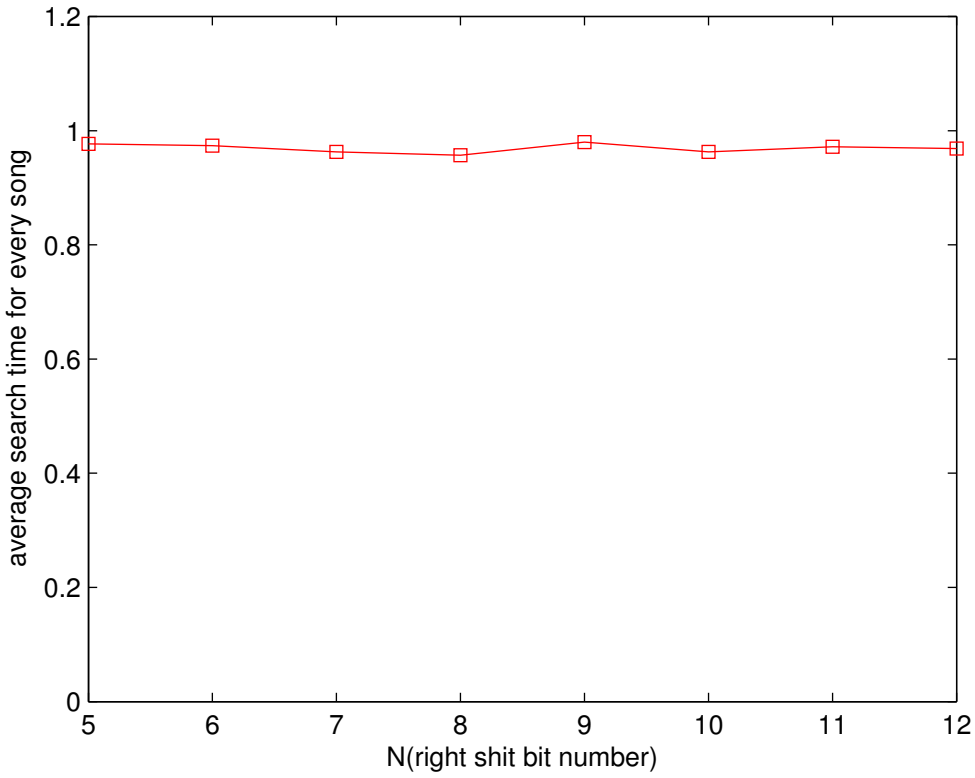

Figure 4.8: Average Usage Rate.

Chapter 5

Similarity Analysis

In this chapter, the goal is to carry out the similarity analysis based on the sub-fingerprint feature. We hope by analyzing similarity to obtain some important information for further understanding and studying repeating patterns.

Repeating patterns have great difference in lengths, so it is very difficult to directly study repeating patterns as a whole. Actually, in literature [29], a granularity of 256 subsequent sub-fingerprints as the basic unit is used to identify a song. That gives us some enlighten effect on solving the problem of the variable lengths in repeating patterns. Therefore, the main idea of our method is to first cut the whole fingerprint sequence into blocks with fixed length and then analyze the similarity distribution of blocks.

We will make study of similarities of fingerprint blocks from three aspects. Firstly, because Jaap Haitsma and Ton Kalker used a large overlap factor to segment audio signal, the similarities for adjacent fingerprint blocks should be very high in theory. Therefore, we first investigate the similarity distribution for adjacent fingerprint blocks. Secondly, we analyze the similarity distribution of the non-similar fingerprint blocks. Finally, we study the similarity distribution of the similar fingerprint blocks. Here the similarity is perceptually similar, namely having similar melody.

Obviously, as mentioned-above, there are three typical relationships between fingerprint blocks: subsequent, non-similar and similar. We design three separate experiments, with respect to three aspects. The training corpus consists of 100 songs performed by both male and female singers and Eq.3.4 is used to calculate the similarity.

5.1 Subsequent Similarities

Because the large overlap is used in extracting features, subsequent fingerprint blocks have a large similarity and are varying as the increasing ID. These adjacent blocks are considered as trivial similar blocks. In order to reduce this redundant, it needs to learn the distribution of similarities in subsequent 31 blocks, namely ID from 1 to 31. Therefore, the objective of the first experiment is to observe the distribution of similarities for subsequent fingerprint blocks. We select all fingerprint blocks of the training corpus, which contain about 2.5 million blocks corresponding to 77.5 million samples. L sets to be 1, 32, 64, 128 and 256, respectively. Finally, we calculate average BER for each ID.

The similarity distribution of subsequent fingerprint blocks is illustrated in 5.1. Some important information can be obtained from Figure 5.1 by analyzing as the following.

Firstly, for four different lengths, the distributions of similarities coincide with each other, so evidently it can illustrate that the similarity distribution of subsequent fingerprint blocks mainly relates to ID and is not associated with block length.

Secondly, an important rule is also shown in Figure 5.1: as the ID increases, the similarity first decreases to the global minimum (BER gets to the global maximum), then increases and finally fluctuates around 0.5. The large similarity in adjacent blocks can also explain the function of the large overlap well.

Moreover, to observe the distribution of similarities when $ID > 31$, we also

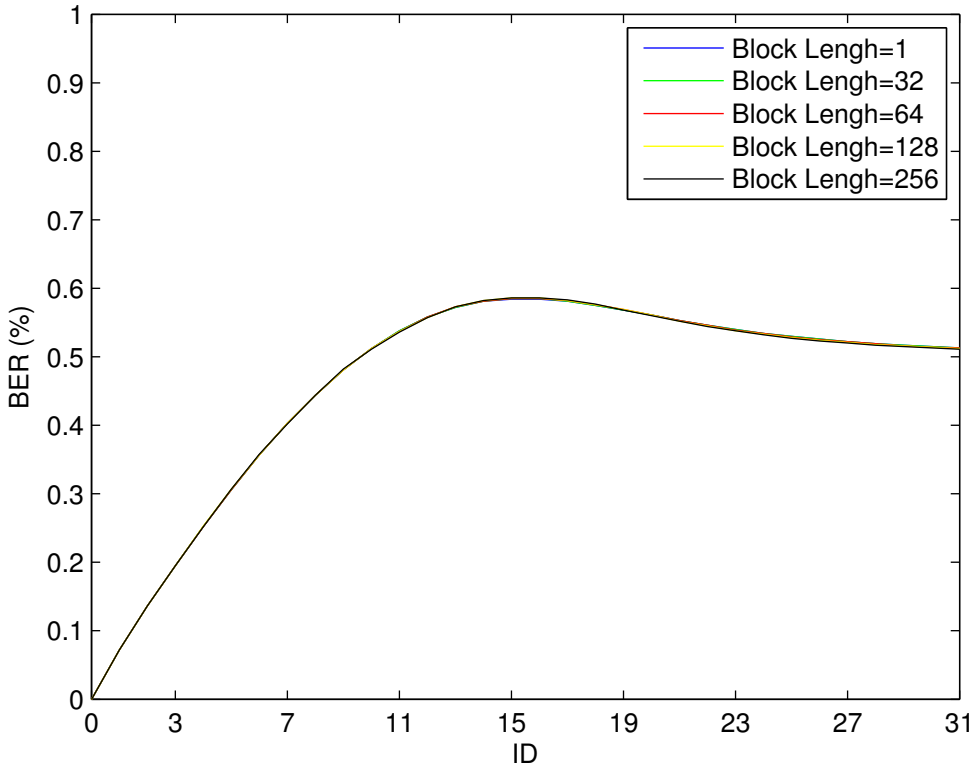

Figure 5.1: Distribution of Subsequent Similarities.

another experiment as the following.

Randomly choosing 20K thousand pairs of fingerprint blocks from each song, and building $20M$ pairs of fingerprint blocks from the training corpus. Moreover, to reduce the impact of subsequent blocks, we also filter meeting the condition $ID \leqslant 31$. In order to fully observer the distribution of similarities, we select five representative lengths: 1, 32, 64, 128 and 256 for experiments. Finally, we calculate the BER scores for these block pairs and observe the score distributions.

When $ID > 31$, the distribution of similarities is shown in Figure 5.2 and this distribution approximately follows the normal distribution. Obviously, as the block length is increasing, the similarities are characterized by intensive distribution.

In this subsection, the experimental results indicate that subsequent blocks have a large similarity.

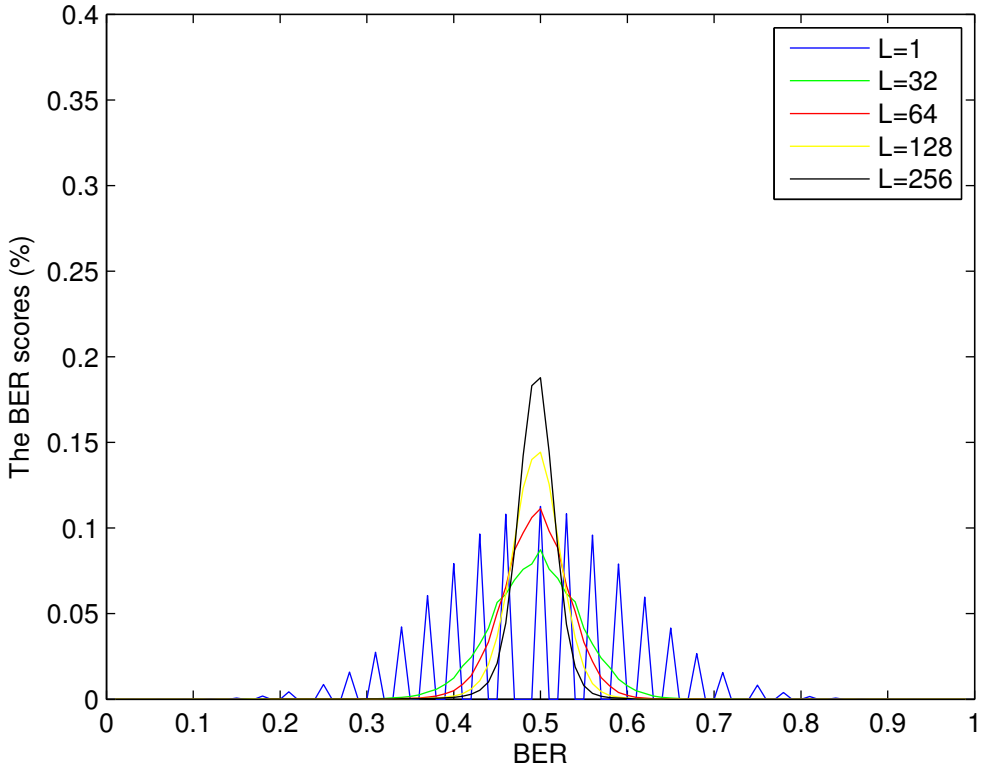

Figure 5.2: Distribution of Random Similarities.

5.2 Non-Similar Similarities

The goal of this subsection is to study the similarity distribution of non-similar fingerprint blocks. Here the non-similar is relative to the similar. How to get non-similar block pairs is the key to success in this experiment. Normally, fingerprint blocks from different songs are less likely to be similar, if two fingerprint blocks of each pair are randomly selected from two different songs, in this case, the probability of the similar can be almost ignored.

Based on these analyses, the second experiment is designed: randomly choosing 2K thousand pairs of fingerprint blocks for every two songs, and building 20M pairs of fingerprint blocks from the training corpus. Moreover, to further observe the distribution of similarities for different block lengths, we also choose five representative lengths: 1, 32, 64, 128 and 256 for experiments. Finally, we calculate the BER

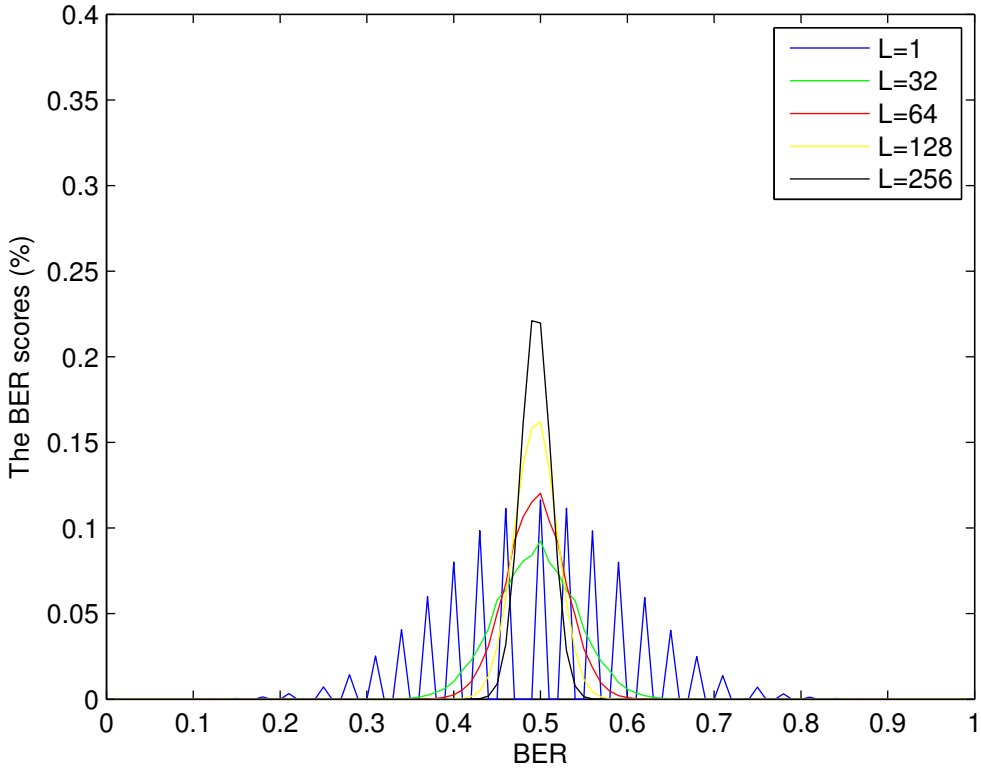

Figure 5.3: Distribution of Non-Similar Similarities.

scores for these block pairs and observe the score distributions.

The distribution of similarities in non-similar blocks also approximately follows the normal distribution as illustrated in Figure 5.3. As a matter of fact, Figure 5.3 is very similar to Figure 5.2. Besides, as the block length is increasing, the distribution is more concentrated, which means that the longer block is more expressive and representative together with Figure 5.2.

In this paper, we mainly focus on $L = 256$. When $L = 256$, the similarity almost lies in the domain from 0.41 to 0.57, with the error of less than 10^{-5}. There are usually about 25 thousands of sub-fingerprints in one song, so $BER = 0.41$ and $BER = 0.57$ can be completely considered as the start and the end of the non-similar fingerprint blocks respectively. In fact, according to definition of bit error bit, the condition $BER > 0.57$ is also non-similar relationship, namely meeting the condition $BER \geq 0.41$ represents the non-similar relationship. Based on these

W	C
1	[0.09,0.9]
32	[0.29,0.69]
64	[0.33,0.63]
128	[0.38,0.60]
256	[0.41,0.56]

Table 5.1: Domain of Non-Similar Similarities.

analyses, the domain of similarities in different lengths is as shown in Table 5.1, with the error of less than 10^{-5}.

This experiment shows that similarities for non-similar fingerprint blocks assume highly localized distributions, especially, as the block length is increasing and the distribution is more centralized.

5.3 Similar Similarities

The third experiment investigates the similarities of similar fingerprint blocks with 256 subsequent sub-fingerprints. Here similar is relative to non-similar, namely $BER < 0.41$. We first design a simple method to find similar blocks and then observe their similarity distribution. The steps are described as follows.

Step 1

A fingerprint sequence $FP = f_1 f_2 ... f_N$ is equally segmented into blocks with 256 subsequent sub-fingerprints and the i-th block is

$$B_i = FP^{256}_{(i-1)\times 256+1}, i \in [1, I], \tag{5.1}$$

Where $I = [\frac{N}{256}]$ is integer part of $\frac{N}{256}$.

Step 2

Each block B_i is detecting its similar blocks in $f_{(i-1)*256+2} ... f_N$.

In testing, we find two types of redundant blocks caused by using the larger overlapping factor in feature extraction:

The first type is that each block B_i is similar to its neighboring blocks. That is

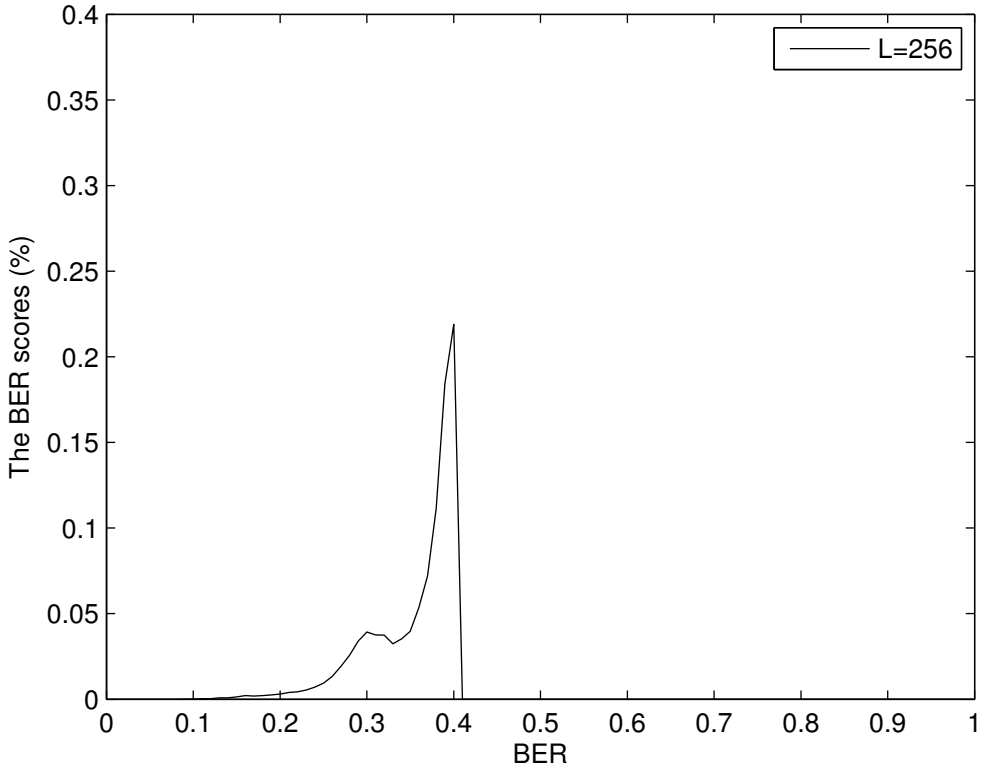

Figure 5.4: Distribution of Similar Similarities.

because the adjacent fingerprint blocks have the larger similarity caused by using the larger overlap factor in extracting. In order to address this redundant, we restrict that each block B_i is related to its similar blocks in $f_{(i-1)*256+}...f_N$. The threshold is defined as the adjacent redundant distance.

The second type is that if both B_i and $B_i^{'}$ are similar, then B_i is also similar to adjacent blocks of $B_i^{'}$. Therefore, after finding $B_i^{'}$, we further check the neighborhood of $B_i^{'}$ to look for a best similar block $B_i^{''}$ in place of $B_i^{'}$. After finishing, let S represent subscript of $B_i^{''}$, then B_i will continue to find its similar blocks in $f_{S+}...f_N$, until the end.

We first employ this algorithm to observe the distribution of similarities in the similar blocks and the threshold sets to be 32. Dataset still comes from the test corpus. By this experiment, these similarities for similar block pairs would be recorded. Experimental results based on the test corpus are illustrated in Figure 5.4.

Figure 5.4 shows that as the BER increases, the BER score first reaches the first peak at $BER = 0.33$, then fast decreases to a minimum at $BER = 0.35$, finally rapidly increases from $BER = 0.37$.

This experiment indicates that the similarities for similar blocks are smaller compared with the non-similar.

Chapter 6

Detecting Repeating Patterns Based on Similarity Analysis

In this chapter, according to similarity analyses of Chapter 5, a relevant algorithm is proposed to discover repeating patterns. Experiments based on a test corpus of 30 familiar songs are used to evaluate the whole algorithm. Finally, we fully analyze performance of our methods.

6.1 Detection Algorithm

In this section, a new algorithm is designed to detect repeating patterns for verifying the similarity analysis of Chapter 5. In literature [45], it calculated similarities of all element-wise to capture similar elements; literature [46] tried to use the global correlation to capture variable-length similar segments based on the element-wise comparison through adaptive thresholds. In fact, Chapter 4 has been proven that the longer block is more expressive and representative, so the block-to-block measure is more suitable for our method. It is apparent that a true repeating pattern is usually consisted of many subsequent blocks. Therefore, we first capture similar blocks, join these similar blocks to form the longer blocks and finally extract repeating patterns by refining the longer blocks. This approach is described as the following

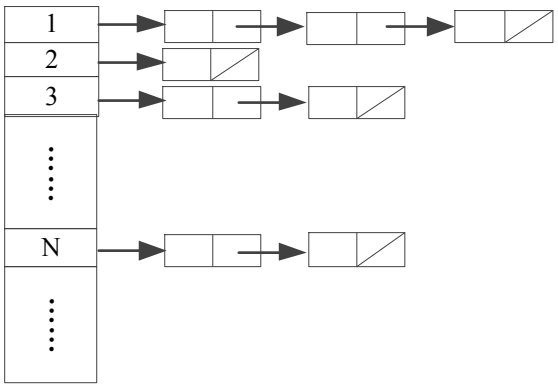

Figure 6.1: Hash Table of Storing Similar Blocks.

subsections.

6.1.1 Capturing Similar Blocks

The aim of this subsection is to capture similar blocks. Based on literature [29], we also consider that a block with 256 subsequent sub-fingerprints can identify a repeating pattern. So the block length sets to be 256 in our method. In addition, the similar threshold is set: $\rho = 0.41$, namely if meeting the condition $BER < 0.41$, it represents similar relationship. After capturing similar blocks, they will be stored into a hash table HT as shown in Figure 6.1, of which each item represents a segmented block B_i and points to a list that stores all the information of all similar blocks. In the list, each node represents a similar block and is consisted of data domain and pointer domain. The data domain contains the first sub-fingerprint subscript and its length of a similar block. The default of block length is equal to be 256.

6.1.2 Mergence

A lot of similar blocks are generated in the above subsection. In this subsection, these similar blocks will be combined to form the longer blocks. Because of using

the high overlapping factor in the feature extraction, it also causes misalignments in subsequent blocks. So the error of 32 sub-fingerprints is allowed in combining.

Mergence Rule

Similar blocks which belong to the same repeating pattern are combined together to form the longer block by modifying length of the first sub-fingerprint and deleting the nodes at which the combined blocks are located.

Mergence Description

If subsequent similar blocks $\theta_1 ... \theta_{i-1}$ are coming from $HT[j]$,...$HT[i-j-2]$ and $HT[i-j-1]$ has a similar block θ_i meeting Eq.6.1:

$$|\theta_i.index - \theta_1.index - \theta_1.length| < 32. \tag{6.1}$$

Then, these blocks $\theta_1 ... \theta_i$ belong to the same pattern. Where θ_1 represents the first sub-fingerprint. Equation 6.2 is used to update length of θ_1 and θ_i is removed from $HT[i-j-1]$.

$$\theta_1.length = \theta_i.index - \theta_1.index + \theta_i.length. \tag{6.2}$$

After merging, we obtain a lot of the longer blocks. But, there are still some of similar blocks which have not been combined. Such blocks mainly come from the noise or slightly similar and are very difficult to distinguish.

In order to reduce other influences, in this paper, we restrict length of repeating patterns to be longer than 512 sub-fingerprints (approximates 6 sec duration). The shorter will be removed.

6.1.3 Boundary Refinement

In our method, mainly using segmented blocks are to capture similar blocks, so there are missing sub-fingerprints in two boundaries of each repeating in HT. It needs to perform the boundary refinement. According to the whole fingerprint sequence and

$\rho = 0.41$, the missing parties are very easily found by traversing along two boundaries of the longer blocks.

After the boundary refinement, in HT, each node in left nodes represents a detected repeating pattern. According to sampling rate, the start and end time are easily computed.

6.2 Experiments and Analyses

In Chapter 5, the similarity analysis is carried out by three experiments. According to similarity analysis, a related method is presented to discover repeating patterns for verifying the analysis results of Chapter 5. This subsection will evaluate the whole algorithm.

Actually, no matter what types of music, the comprehension for repeating patterns is the same. So these songs with clear structure and relatively strict repetition should be used to test for improving the quality of experiments. These works [45,46] have shown that pop music usually has obvious repeating structures and is very easily to test. In our tests, a test corpus of 30 familiar pop songs is used. We also annotate the ground truth of the repeating patterns on these 30 songs by training a lot and these annotated patterns are exploited as our ground truth patterns. In the annotation, we only consider such repeating patterns, with a length longer than 6 s.

6.2.1 Evaluation on Algorithm

In this chapter, the proposed method is based on similarity analysis. In this subsection, we will test whether it is correct and use three criteria: recall, precision and $F1$ to evaluate.

According to sample rate, time representation of the annotated patterns is first converted into the sub-fingerprint representation. The recall and precision of each repeating pattern are calculated based on the number of sub-fingerprints, with refer-

	Recall	Precision	$F1$
Our Method	81.7%	81.7%	79.9%

Table 6.1: Evaluation on Algorithm.

ence to subscripts. To achieve a robust evaluation, the error with 8 sub-fingerprints is allowed in between the annotated patterns and the detected patterns.

If S denotes the set of detected repeating patterns generated from Section 6.1 and Q for the ground truth patterns of each song, then the correctly detected repeating patterns are represented by $S^{'}$:

$$S^{'} = S \cap Q. \tag{6.3}$$

Where $\cap$ denotes the intersection operator, namely including the same sub-fingerprints, with reference to subscripts. In this paper, the recall R is expressed as:

$$R = \frac{S^{'}}{Q}. \tag{6.4}$$

The precision P is described as:

$$P = \frac{S^{'}}{S}. \tag{6.5}$$

The $F1$ measure represents the overall performance, which is usually defined as the harmonic mean of the average recall and precision:

$$F1 = \frac{2 \times R \times P}{R + P}. \tag{6.6}$$

The average recall, precision and $F1$-measure based on these 30 songs are illustrated in Table 6.1.

Table 6.1 shows how our approach can discover the true repeating patterns well. It shows that exploiting segmented blocks instead of the whole repeating patterns

is feasible. Besides, the setting of similar threshold $BER = 0.41$ is reasonable, namely if meeting the given conditions, the similarity distributions of both repeating patterns and non-repeating pattern are approximately distinguishable in fingerprint features.

6.2.2 Analysis on Performance

In Section 6.2.1, it has shown that our method based similarity analysis is feasible. In this subsection, we will evaluate its performance again. Because the longer block is more expressive and representative, the block-to-block similarity measure should be more efficient. We here also employ the fourth and fifth experiments to compare its performance with two relevant approaches: self-similarity [45] and AMG [46]. In these three methods, they all focused on melody, namely repeating patterns represent having similar melodies. Besides, their features are also similar based on perceptual feature.

The fourth experiment: computing the element-wise similarities ($L = 1$) in detected repeating patterns and observing the distribution of similarities. The results are shown in Figure 6.2.

The fifth experiment: randomly selecting a pair of the similar blocks (the $BER = 0.29$) from detected repeating patterns to compute the element-wise similarities and observing similarities of subsequent element-wises. The results are shown in Figure 6.3.

The element-wise similarity mainly lies in the domain from 0.05 to 0.8 based on the test corpus as shown in Figure 6.2. In each repeating pattern, the element-wise similarity greatly varies and the change of subsequent element-wise similarities is disorder as illustrated in Figure 6.3. Therefore, literature [45] used a single threshold to capture similar data and literature [46] tried to use adaptive thresholds and suffix tree to capture variable similar blocks based on the global correlation, and clearly the ability of repeating pattern recognition is low. That leads to heavily

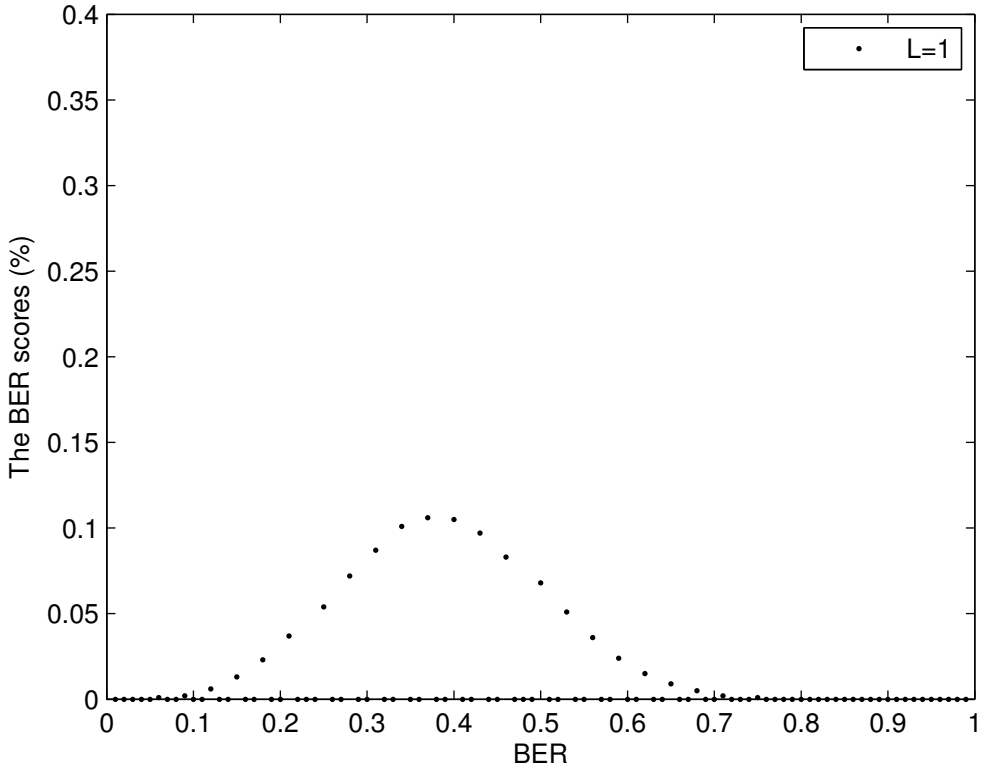

Figure 6.2: Element-wise Similarities in Repeating Patterns.

relying on refinements to extract repeating patterns in these two approaches. In our method, the block-to-block similarity measure is applied and the longer block is more expressive and representative as shown in Figure 5.3, obviously the ability for recognizing repeating patterns is remarkable. Therefore, similar blocks are easily identified and works of extracting repeating patterns are more efficient compared with [45] and [46].

6.2.3 Analysis on Ability of Recognition

In this subsection, we further analyze the ability of recognition of repeating patterns. To evaluate, we also propose three standards: duration, the average similarity and the discovery rate of repeating patterns to measure the ability of recognition, respectively representing in T, AS and DR. T represents duration of each repeating pattern. The AS represents the overall similarity of each repeating pattern and

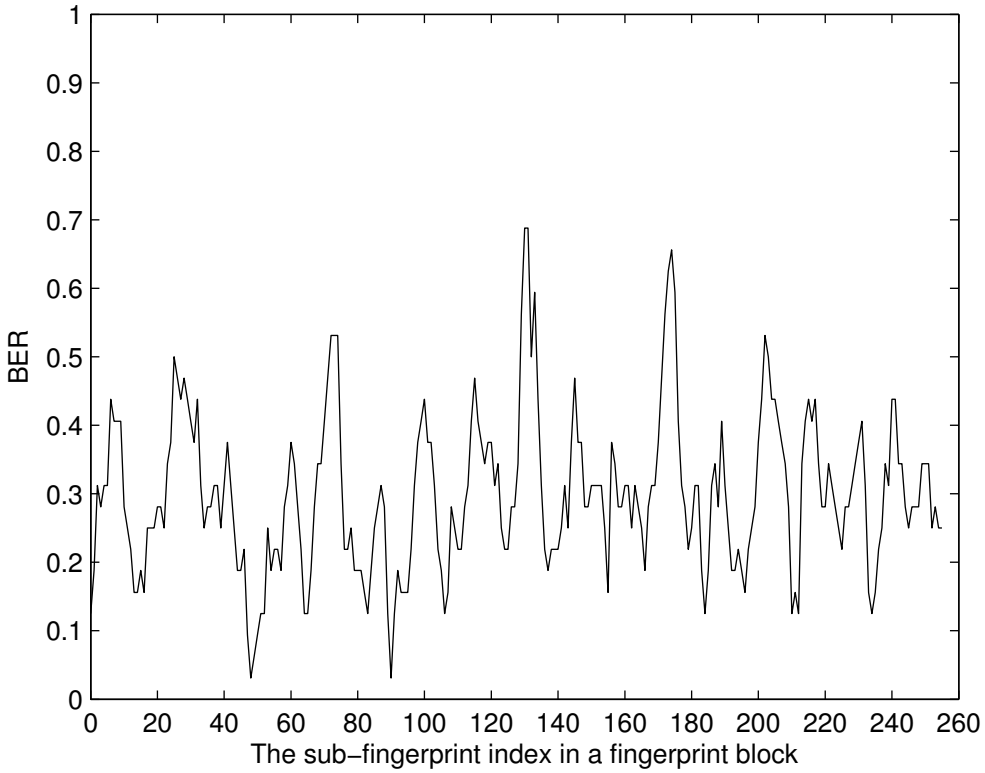

Figure 6.3: Element-wise Similarities in A Similar Block Pair.

we here compute the average similarity of all similar blocks to represent AS. The DR shows the ratio of all real similar blocks to the detected similar blocks in each repeating pattern.

For a given repeating pattern, if it is composed of n similar blocks, let BER_i represent similarity of i-th similar blocks; g represents the amount of missing blocks compared to real repeating patterns. Then AS and DR are defined as follows.

$$AS = \frac{\sum_{i=1}^{n} BER_i - G}{n - g}. \tag{6.7}$$

$$DR = \frac{n - g}{n}. \tag{6.8}$$

For the missing blocks, the similarity sets to be 0. The longer the T, the better the robustness of our approach is; the smaller the value AS, the higher similarities

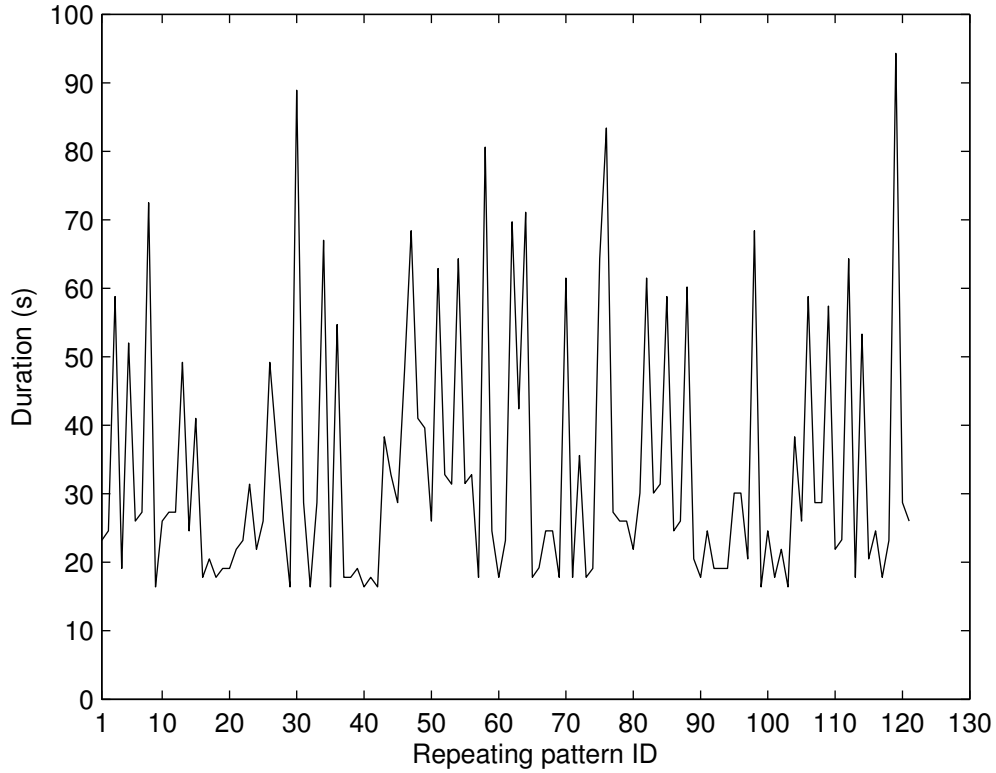

Figure 6.4: Duration of Repeating Pattern.

are; while the bigger the values DR, the higher similarities are. Therefore, T, AS and DR can reflect the ability of pattern recognition.

Here we employ the test corpus to measure and finally get 121 repeating patterns and the results are as shown in Figure 6.4, Figure 6.5 and Figure 6.6.

The ability of repeating recognition is as illustrated in Figure 6.4, Figure 6.5 and Figure 6.6. Through our method, based on the test corpus, the average duration 33.4s is identified as shown in Figure 6.4. Besides, we can see that lengths of repeating patterns greatly vary. Figure 6.5 shows the overall similarity of each repeating pattern and the similarity is obviously less than the non-repeating. From Figure 6.6, we can observe the ratio of missing sub-fingerprints in each real repeating pattern. In some of repeating patterns, the DR approximately achieves 100% and we believe that these patterns should have clear repetitive structure. From three standards, we can clearly observe the recognition capability of our method.

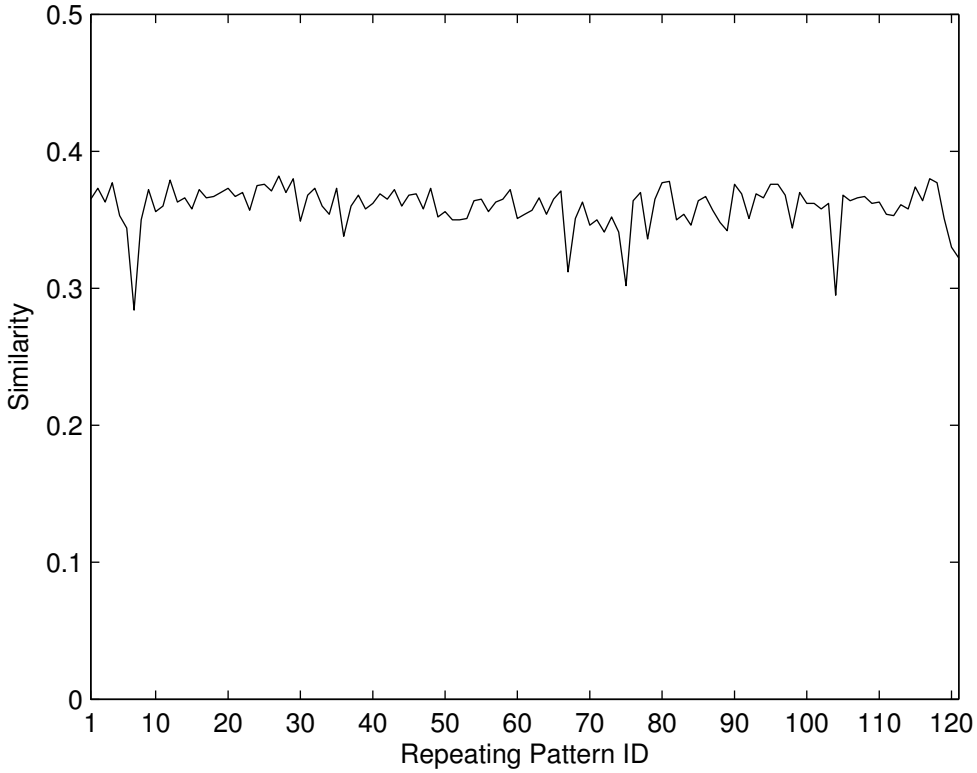

Figure 6.5: Overall Similarity of Repeating Pattern.

Next we use these 121 repeating patterns to observe the similarity distribution of similar block pairs in repeating patterns. The experimental results are plotted in Figure 6.7.

The distribution of similarities in repeating patterns is as indicated in Figure 6.7. By comparing Figure 5.4 and Figure 6.7, the similarity in the repeating patterns is better than the similar blocks. So that means that repeating patterns are mainly coming from similar blocks. Besides, it can also explain that exploiting blocks as repeating patterns is feasible.

6.2.4 Analysis on Structure

Due to use the high overlapping factor in the feature extraction, it leads to occurring misalignments in subsequent blocks. To handle this problem, in our method, the error of 32 sub-fingerprints is allowed in merging. Here we further analyze this

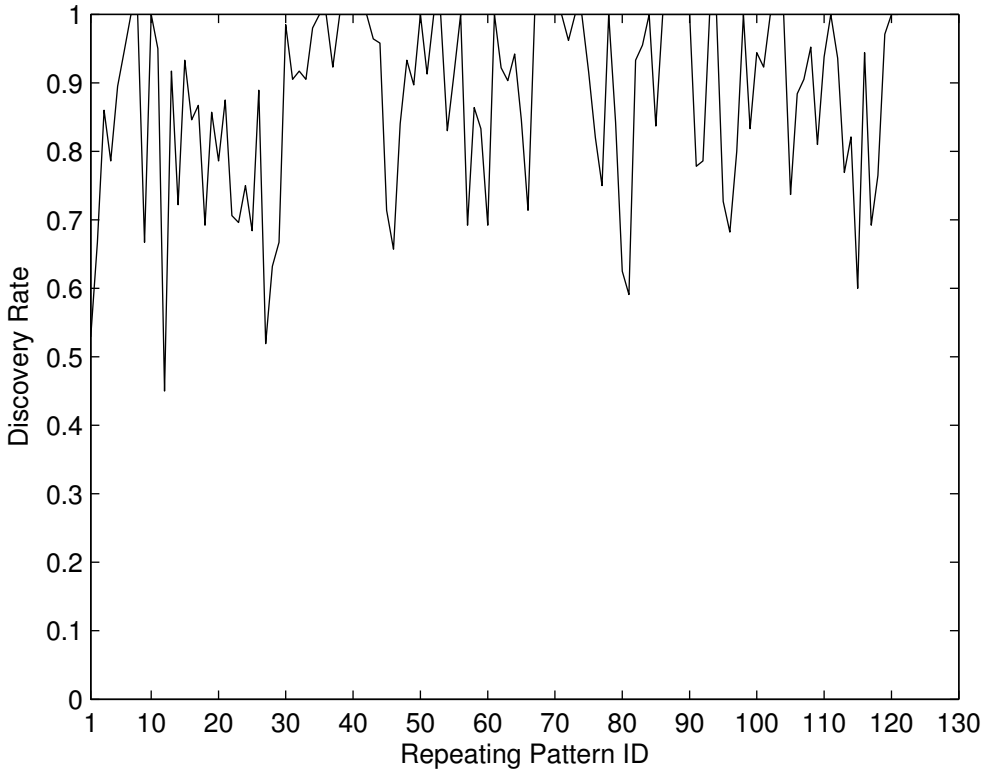

Figure 6.6: Discovery Rate of Repeating Pattern.

settlement whether it is reasonable. We exploit the test corpus to perform the test. The final experiments are as revealed in Figure 6.8.

The error is mainly centering at (-9, 6) about 98.7% as illustrated in Figure 6.8. In our method, after capturing similar blocks, we further select a best similar block from its subsequent blocks to merger. So it can ensure that the optimum of similar blocks is identified. From experimental results, that method is feasible and the condition of the error of 32 sub-fingerprints is correct. Besides, this experiment can also explain as one reason of the missing sub-fingerprints.

6.2.5 Analysis on Complexity

In this chapter, based on similarity analysis, we present a new method to detect repeating patterns. Here we want to show that efficiency of this method. We mainly analyze the time and space complexity and compare with self-similarity [45]

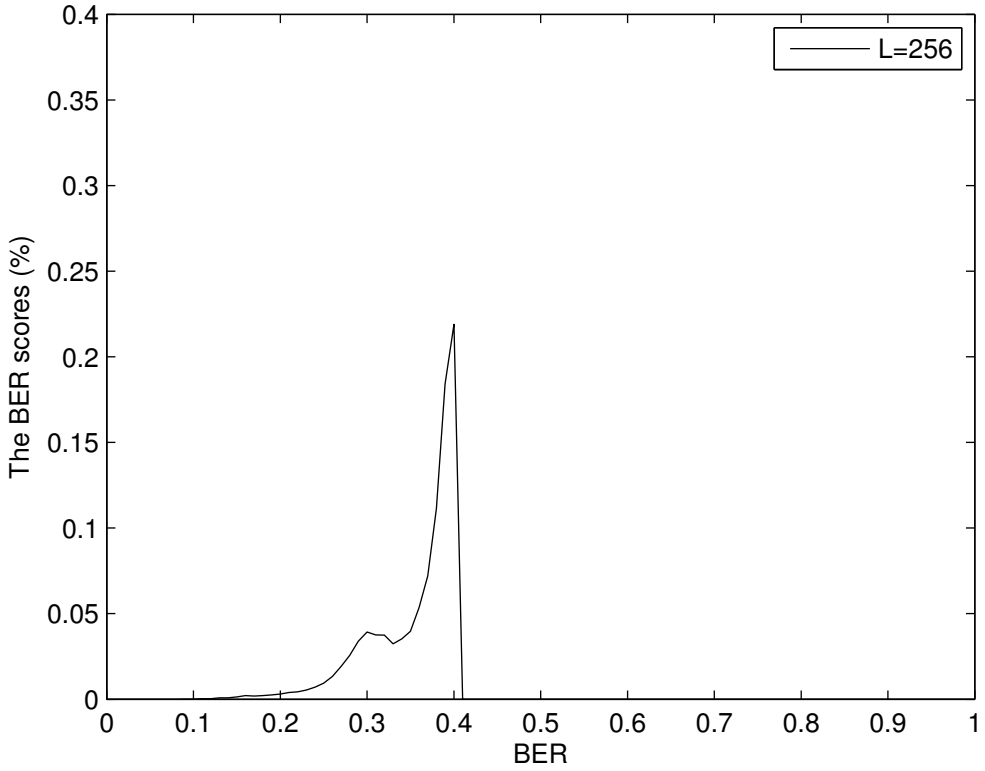

Figure 6.7: Similarity of Repeating Pattern.

and AMG [46].

The space complexity

In detecting repeating patterns, the whole process contains two phases: capturing similar elements or blocks and further refining them to extract repeating patterns. In the first phase, the difference of these three methods is not obvious and we mainly focus on the second phase.

Firstly, we evaluate the space complexity by memory usage. In our algorithm, after capturing similar blocks, they are stored to a hash table with a size of $\frac{N}{256}$, where N represents the total amount of sub-fingerprints. Let P represent the number of similar blocks obtained from Section 6.1, thus the complexity of space is described:

$$O(2 \times Q + 2 \times P) \approx O(2 \times P) < O(N). \tag{6.9}$$

In self-similarity [45] and AMG [46], they need to construct a $o(N \times N)$ self-

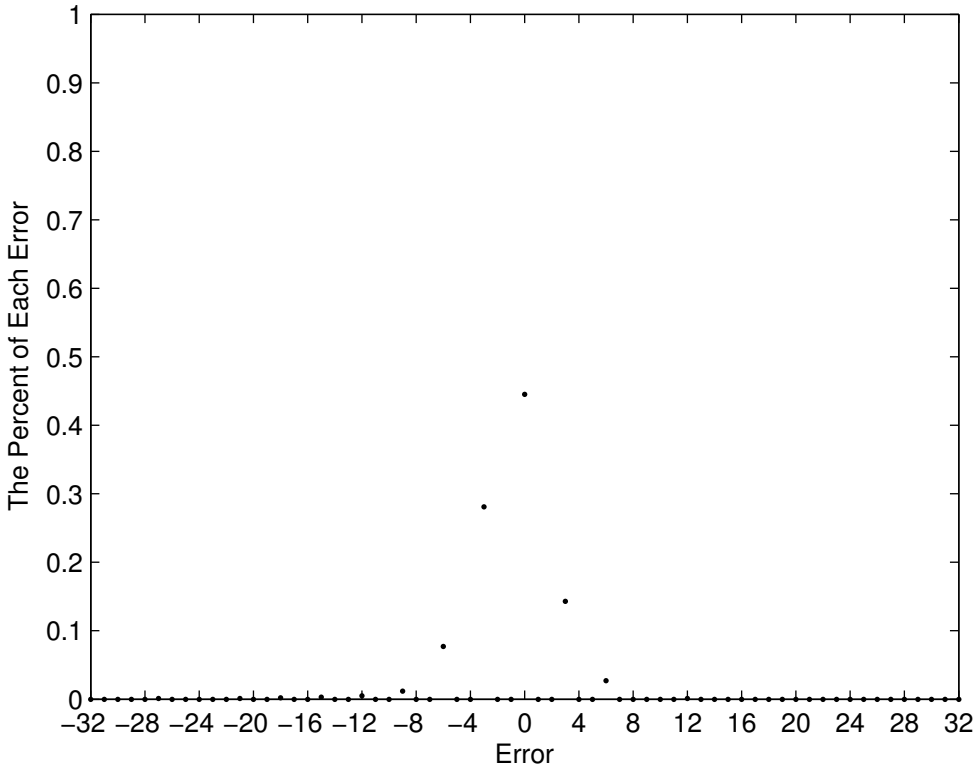

Figure 6.8: Distribution of Misalignment Error.

similarity matrix to store similar elements or the candidate motifs for further extracting repeating patterns.

The time complexity

Next we analyze time complexity. In our method, we use a hash table with a size of $\frac{N}{256}$ to record all similar blocks. And the average number of nodes in the lists is equal to 6 based on the test corpus. Therefore, we fairly extract the repeating patterns in a $o(N \times 6)$ matrix; while for the self-similarity and AMG, it needs to extract repeating patterns in a sparse $o(N \times N)$ matrix. Obviously, our approach is better than self-similarity and AMG.

Through analysis of complexity, our improvement is obvious in space and time complexity, compared to the self-similarity and AMG.

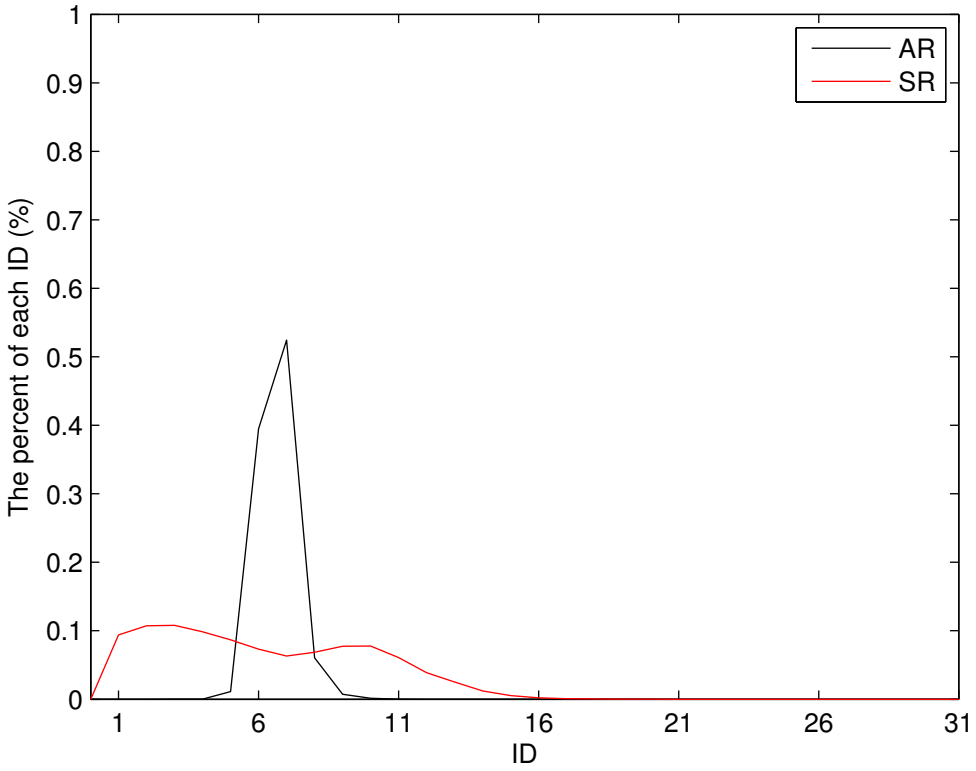

Figure 6.9: Distribution of Redundancy Data.

6.2.6 Analysis on Redundancy Data

In our method, the sub-fingerprint extraction uses a large overlap factor to offset the error of frame boundaries, which causes the high similarities between adjacent fingerprint blocks. So there are two types of redundant blocks are found. The first type is that the current block B_i is similar to its neighboring blocks and called the adjacent redundancy (AR). The second type is that if $B_i^{'}$ is a similar block of the current block B_i, then B_i is also similar to adjacent blocks of $B_i^{'}$. And this redundancy is called similar redundancy (SR).

We here use the test corpus to observe the distributions of two redundancies and the results are as shown in Figure 6.9.

The distributions of two redundancies are as illustrated Figure 6.9. It is noted that the AR is more than the SR. These two redundancies would generate a lot of trivial similar blocks, so they must be filtered. For AR, after capturing similar

blocks, it will jump 32 subs-fingerprints to continue with comparing. For SR, we will select a best similar block near a similar block to combine.

6.3 Summary

This chapter presents a new approach to detect repeating patterns based on the similarity analysis in the sub-fingerprint feature sequence. We firstly segment a sub-fingerprint sequence into blocks and analyze similarities of block pairs. And according to similarity analysis, a relevant method is proposed to detect repeating patterns. The experimental results show that our approach can capture the true repeating patterns well. Moreover, the performance analyses show that our method is prospective.

In future works, we would further study the rules of distribution in similarities of repeating patterns and improve the detection accuracy.

Chapter 7

Contribution and Futher Works

7.1 Conclusion

This thesis studies content-based music retrieval and mainly focus on repeating patterns. In our method, the sub-fingerprint feature is exploited. Our research contains two parts.

In the first part, we present a fast music retrieval method as extension of Philips' method to address problems of limited memory. In our method, Fibonacci Hashing Function is used to provide a good distribution of hash values; the right shift operation is carried out to adjust size of lookup table according the practical memory. The results show that our method is more robust and has a wide range of adaptability.

In the second part, we study similarities of repeating patterns and propose a relevant approach to detect repeating patterns based the similarity analysis. In analyzing, in order to solve the variable lengths of repeating patterns, blocks instead of repeating patterns are studied. Final experiments show that repeating patterns can be detected well in our method, so our method is feasible.

7.2 Futher Researches

Firstly, in Chapter 4, because the right shift operation is carried out, for the high distortion, we need to consider more candidate sub-fingerprints compared to Philips' method. So it is drawback of our method. Next we consider by exploiting similarity analysis to study this problems

Secondly, in Chapter 6, we only use sub-fingerprint feature to study repeating patterns. We believe that this method should also be applied to other features. Next we try to verify. Besides, we also further study rules of repeating patterns in distribution of similarities to obtain more useful information.

Bibliography

[1] N. Orio. *Music retrieval: A tutorial and review*, volume 1. 2006.

[2] [2] S. L. Vellucci. Music metadata. 2004.

[3] Jonathan Foote. An overview of audio information retrieval. *Multimedia systems*, 7(1):2–10, 1999.

[4] George Tzanetakis and Perry Cook. Audio information retrieval (air) tools. In *Proc. International Symposium on Music Information Retrieval*, 2000.

[5] Rainer Typke, Frans Wiering, Remco C Veltkamp, et al. A survey of music information retrieval systems. In *ISMIR*, pages 153–160, 2005.

[6] Dalibor Mitrović, Matthias Zeppelzauer, and Christian Breiteneder. Features for content-based audio retrieval. *Advances in computers*, 78:71–150, 2010.

[7] Martin F McKinney and Jeroen Breebaart. Features for audio and music classification. In *ISMIR*, volume 3, pages 151–158, 2003.

[8] Mingchun Liu and Chunru Wan. A study on content-based classification and retrieval of audio database. In *Database Engineering and Applications, 2001 International Symposium on.*, pages 339–345. IEEE, 2001.

[9] Michael Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, Malcolm Slaney, et al. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[10] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19, 2006.

[11] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19, 2006.

[12] Cheng Yang. Music database retrieval based on spectral similarity. 2001.

[13] Jyh-Shing Roger Jang and Hong-Ru Lee. Hierarchical filtering method for content-based music retrieval via acoustic input. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 401–410. ACM, 2001.

[14] Yazhong Feng, Yueting Zhuang, and Yunhe Pan. Popular music retrieval by detecting mood. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 375–376. ACM, 2003.

[15] Jonathan Foote, Matthew L Cooper, and Unjung Nam. Audio retrieval by rhythmic similarity. In *ISMIR*, 2002.

[16] Lie Lu, Hong You, HongJiang Zhang, et al. A newapproach to query by humming in music retrieval. In *ICME*, pages 22–25, 2001.

[17] Malcolm Slaney. Semantic-audio retrieval. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 4, pages IV–4108. IEEE, 2002.

[18] Michael I Mandel, Graham E Poliner, and Daniel PW Ellis. Support vector machine active learning for music retrieval. *Multimedia systems*, 12(1):3–13, 2006.

[19] Jonathan T Foote. Content-based retrieval of music and audio. In *Voice, Video, and Data Communications*, pages 138–147. International Society for Optics and Photonics, 1997.

[20] Yuen-Hsien Tseng. Content-based retrieval for music collections. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 176–182. ACM, 1999.

[21] Peter Grosche, Meinard Müller, and Joan Serrà. Audio content-based music retrieval. *Multimodal Music Processing*, 3:157–174, 2012.

[22] Dongge Li, Ishwar K Sethi, Nevenka Dimitrova, and Tom McGee. Classification of general audio data for content-based retrieval. *Pattern recognition letters*, 22(5):533–544, 2001.

[23] Tong Zhang and C-C Jay Kuo. Content-based classification and retrieval of audio. In *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pages 432–443. International Society for Optics and Photonics, 1998.

[24] Erling Wold, Thom Blum, Douglas Keislar, and James Wheaten. Content-based classification, search, and retrieval of audio. *MultiMedia, IEEE*, 3(3):27–36, 1996.

[25] Pedro Cano. *Content-based audio search: from fingerprinting to semantic audio retrieval*. PhD thesis, Citeseer, 2006.

[26] Pedro Cano, E Batle, Ton Kalker, and Jaap Haitsma. A review of algorithms for audio fingerprinting. In *Multimedia Signal Processing, 2002 IEEE Workshop on*, pages 169–173. IEEE, 2002.

[27] Pedro Cano, Eloi Batlle, Emilia Gómez, Leandro de CT Gomes, and Madeleine Bonnet. Audio fingerprinting: concepts and applications. In *Computational intelligence for modelling and prediction*, pages 233–245. Springer, 2005.

[28] Avery Wang et al. An industrial strength audio search algorithm. In *ISMIR*, pages 7–13, 2003.

[29] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *ISMIR*, volume 2002, pages 107–115, 2002.

[30] `http://www.music-ir.org/mirex/wiki/2015:Discovery_of_Repeated_Themes_%26_Sections/`.

[31] Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[32] `[32]https://en.wikipedia.org/wiki/K-means_clustering/`.

[33] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[34] Ning-Han Liu, Yi-Hung Wu, and Arbee LP Chen. An efficient approach to extracting approximate repeating patterns in music databases. In *Database Systems for Advanced Applications*, pages 240–252. Springer, 2005.

[35] Jia-Lien Hsu, Chih-Chin Liu, and Arbee LP Chen. Discovering nontrivial repeating patterns in music data. *Multimedia, IEEE Transactions on*, 3(3):311–325, 2001.

[36] Ioannis Karydis, Alexandros Nanopoulos, and Yannis Manolopoulos. Finding maximum-length repeating patterns in music databases. *Multimedia Tools and Applications*, 32(1):49–71, 2007.

[37] Chaokun Wang, Jianzhong Li, and Shengfei Shi. N-gram inverted index structures on music data for theme mining and content-based information retrieval. *Pattern recognition letters*, 27(5):492–503, 2006.

[38] Chih-Chin Liu, Jia-Lien Hsu, and Arbee LP Chen. Efficient theme and nontrivial repeating pattern discovering in music databases. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 14–21. IEEE, 1999.

[39] Jean-Julien Aucouturier and Mark Sandler. Finding repeating patterns in acoustic musical signals: Applications for audio thumbnailing. In *Audio Engineering Society Conference: 22nd International Conference: Virtual, Synthetic, and Entertainment Audio*. Audio Engineering Society, 2002.

[40] Wei Chai and Barry Vercoe. Structural analysis of musical signals for indexing and thumbnailing. In *Digital Libraries, 2003. Proceedings. 2003 Joint Conference on*, pages 27–34. IEEE, 2003.

[41] Musataku Goto. A chorus-section detecting method for musical audio signals. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5, pages V–437. IEEE, 2003.

[42] Matthew L Cooper and Jonathan Foote. Automatic music summarization via similarity analysis. In *ISMIR*, 2002.

[43] Lie Lu and Hong-Jiang Zhang. Automated extraction of music snippets. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 140–147. ACM, 2003.

[44] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455. IEEE, 2000.

[45] Lie Lu, Muyuan Wang, and Hong-Jiang Zhang. Repeating pattern discovery and structure analysis from acoustic music data. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 275–282. ACM, 2004.

[46] Lei Wang, Eng Siong Chng, and Haizhou Li. A tree-construction search approach for multivariate time series motifs discovery. *Pattern Recognition Letters*, 31(9):869–875, 2010.

[47] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–498. ACM, 2003.

[48] Jessica Lin Eamonn Keogh Stefano Lonardi and Pranav Patel. Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53–68, 2002.

[49] Jia-Lien Hsu, Arbee LP Chen, and C-C Liu. Efficient repeating pattern finding in music databases. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 281–288. ACM, 1998.

[50] Jouni Paulus and Anssi Klapuri. Music structure analysis by finding repeated parts. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 59–68. ACM, 2006.

[51] Ivan Bjerre Damgård. A design principle for hash functions. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 416–427. Springer, 1990.

[52] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making uowhfs practical. In *Advances in CryptologyCRYPTO'97*, pages 470–484. Springer, 1997.

[53] F Aydin and G Dogan. Development of a new integer hash function with variable length using prime number set. *Balkan Journal of Electrical & Computer Engineering*, 1(1):10–14, 2013.

[54] RL Duncan. Application of uniform distribution to the fibonacci numbers. *The Fibonacci Quarterly*, 5(2):137–140, 1967.

[55] L Kuipers. Remark on a paper by rl duncan concerning the uniform distribution mod 1 of the sequence of the logarithms of the fibonacci numbers. *Fibonacci Quart*, 7(465-466):473, 1969.

[56] George Markowsky. Misconceptions about the golden ratio. *The College Mathematics Journal*, 23(1):2–19, 1992.

[57] George Markowsky. Misconceptions about the golden ratio. *The College Mathematics Journal*, 23(1):2–19, 1992.

[58] John HE Cohn. Square fibonacci numbers, etc. *Fibonacci Quart*, 2:109–113, 1964.

[59] Godfrey H Hardy. The theory of numbers. *Science*, pages 401–405, 1922.

[60] D Knuth. The art of computer programming 1: Fundamental algorithms 2: Seminumerical algorithms 3: Sorting and searching, 1968.