Efficient Repeating Pattern Finding in Music Databases*

Jia-Lien Hsu, Chih-Chin Liu, and Arbee L. P. Chen
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: alpchen@cs.nthu.edu.tw

ABSTRACT

In this paper, we propose an approach for the extraction of the repeating patterns in music objects. A repeating pattern is a sequence of notes which appears more than once in a music object. It is one of the most important music features which can be used for both content-based retrieval of music data and music data analysis. We propose a data structure called correlative matrix and its associated algorithms for extracting all repeating patterns in a music object. Experiments are also performed and the results are analyzed to show the efficiency and the effectiveness of our approach.

Keywords: music databases, content-based music data retrieval, music feature extraction, music representation, repeating patterns

1. INTRODUCTION

In recent years, the searching and indexing techniques for multimedia data are getting more attention in the area of multimedia databases. As many research works were done on the content-based retrieval of image and video data [7][8][11][16][20][24], less attention was received to the content-based retrieval of audio data. Traditionally, the audio objects are treated as a large sequence of bytes. Except for some descriptive attributes such as name, file format, or sampling rate, an audio object is treated as an opaque large object in which its semantics is omitted. Although many meaningful feelings can be comprehended by human beings when listening to an audio object, it is very difficult to automatically extract these feelings from the raw data of the audio object.

The audio objects can be classified into two groups: music and sound objects, according to whether they have associated *staffs*. For the audio objects, Wold *et al.* proposed an approach to retrieve them based on their content [26]. In their approach, an N-vector is constructed according to the acoustical features of an audio object. These acoustical features include *loudness*, *pitch*, *brightness*, *bandwidth*, and *harmonicity*, which can be automatically computed from the raw data. The N-vector of acoustical features is then used to classify sounds for similar searching. However, the acoustical features are at a level too low for the human beings. The most straightforward way to query the music databases for a naive user is to hum a piece of music as the query example to retrieve similar music objects. This approach is adopted in [5][6][10][17]. Ghias *et al.* proposed an approach to transform a music query example into a string which consists of three kinds of symbols ("U", "D", and "S" which represent a note is higher, lower, or the same as its previous note, respectively) [10]. The problem of music query processing is then transformed into that of approximate string matching. However, only using three kinds of symbols is too rough to represent the melody of a music object. Moreover, the proposed string matching algorithm did not take music characteristics into consideration.

To develop a content-based music database system, we have started a project called *Muse* [5][6][17]. In this project, we have proposed three different methods for the content-based music data retrieval. In [17], we treat the rhythm, melody, and chords of a music object as a music feature string and develop a data structure called 1D-List to efficiently perform approximate string matching. Similarity measures in the approximate string matching algorithm are designed based on the music theory. In [6], we consider music objects and music queries as sequences of chords. An index structure is developed to provide efficient partial matching ability. In [5], we propose an approach for retrieving music objects by rhythm. In this approach, the rhythm of a music object is modeled by rhythm strings consisting of *mubols*. A mubol is the rhythmic pattern of a measure in a music object. Five kinds of similarity relationships between two mubols are defined and the similarity measure of two rhythm strings can be calculated

CIKM 98 Bethesda MD USA

*This work was partially supported by the Republic of China National Science Council under Contract No. NSC 88-2213-E-007-052

based on these similarity relationships. An index structure is also proposed for efficiently matching rhythm strings.

No matter the content-based music data retrieval is based on humming, melody, chord, or rhythm, string matching is the basic technique for the music query processing. However, if the music objects are large (for example, 1000 notes), the execution time for query processing will become unacceptable. By examining the staff of a music object, we find many sequences of notes, called *repeating patterns*, appear more than once in the music object. For example, the well known "sol-sol-sol-mi" in Beethoven Symphony No. 5 repeatedly appear in the music object. Many researchers in musicology and music psychology fields also agree that repetition is a universal characteristic in music structure modeling [2][13][19][25]. Therefore, to represent a music object by its repeating patterns instead of the whole music object meets both efficient and semantic requirements for content-based music data retrieval. An efficient technique for finding the repeating patterns from the sequence of notes of a music object is thus needed to be developed.

In this paper, we propose an approach to efficiently find all repeating patterns of music objects. A data structure called *correlative matrix* is used to keep the intermediate results during the finding process.

In addition to supporting the content-based music data retrieval, the repeating patterns can also be used in music data analysis. For example, we can cluster music data based on the similarity of their repeating patterns. Another example is that we can analyze a certain composers' music works to see whether they share a common set of repeating patterns.

The paper is organized as follows. In Section 2, we analyze various music features and explain the semantics of the repeating patterns. We use an example to show the basic idea of the proposed method for finding repeating patterns in Section 3. Some experiments are performed to show the efficiency of the proposed algorithms. The results are further analyzed to show the effectiveness of the approach in Section 4. Finally, Section 5 concludes this paper and presents our future research directions. The algorithms for finding repeating patterns are omitted from this paper due to space limitation.

2. The Role of Repeating Patterns in Music Objects

In this section, we first discuss various music features which can be used to represent the music objects. Then we explain the semantics of the repeating patterns.

For content-based music data retrieval, we need first to extract music features from the raw data of the music objects and store them as the music index for further processing. According to the characteristics of music, the music features can be classified into three categories: *static music information*, *acoustical features* and *thematic features*.

- Static music information refers to the intrinsic music characteristics of music objects such as *key*, *beat*, and *tempo*. For example, the Beethoven Symphony No. 5, Op. 67, is in c minor, 4/4, Allegro con brio. This information can be regarded as keywords or attributes of a music object.
- Acoustical features include loudness, pitch, duration, bandwidth and brightness, which can be automatically computed from the raw data of music objects. For example, the loudness of a music object is the root-mean-square value in *decibels* of its audio signals. The acoustical features of a music object can be represented as a feature vector or feature point in a multi-dimensional feature space.
- Thematic features such as themes, melodies, rhythms, and chords are derived from the staff information of a music object. A theme is the most expressive and representative part of a music object. Generally, it appears repeatedly within a music object. A theme consists of a melody and a rhythm. Thematic features can be represented in string form. For example, "sol-sol-sol-mi" is a melody string, "0.5-0.5-0.5-2" is a rhythm string, and "C-Am-Dm-G7" is a chord string. The melody strings, the rhythm strings, and the chord strings can be easily derived from the music objects in MIDI form [18].

Although it is easy for the naive users to specify the query conditions such as "composer = Beethoven" or "key = c minor", many music features are inadequate to be stored in keyword form. On the other hand, the semantics of the acoustical features of music objects is not obvious to the naive users. For example, it is not common for users to pose a query such as "retrieve all music objects whose bandwidths are similar to". Therefore, we think the thematic features are more essential properties of music objects than the acoustical features.

The classic music objects are composed according to a special structure called *musical form* in which there are two basic rules: hierarchical rule and repetition rule [12][13][19]. The hierarchical rule means music objects are formed hierarchically, *i.e.*, a music object consists of *movements*, a movement consists of *sentences*, a sentence consists of *phrases*, and a phrase consists of *figures*. For example, Figure 1 shows the music form of Brahms Waltz in A flat. It contains one sentence, *i.e.*, sentence 1. Sentence 1 consists of phrase 1 and phrase 2. Both phrase 1 and phrase 2 consist of four figures. These figures, phrases, and sentence form a hierarchical structure.

The repetition rule means that some sequences of notes, known as *motives*, will repeatedly appear in a movement. For example, both of the sequences "C6-Ab5-Ab5-C6" and

"F6-C6-C6-Eb6" repeat two times in Figure 1 and their rhythms are of the same, which is "1.5-0.5-0.5-0.5".

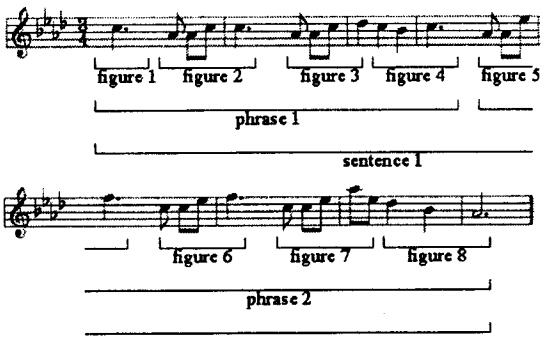
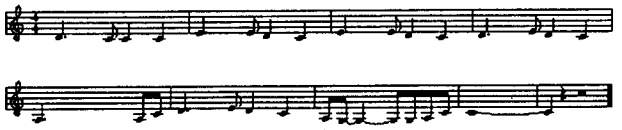


Figure 1: Brahms Waltz in A flat.

The repetition rule can also be found in pop music. For example, the refrain of the song "Five Hundred Miles", as



shown in Figure 2 , repeats three times in this song.

Figure 2: The refrain of the song "Five Hundred Miles".

The refrains in a pop song and the motives of the classic music are typical themes. Since themes play important roles in the music objects, they can be adopted as the main music features for content-based music data retrieval. To achieve this, we need to develop an efficient technique to automatically find the motives or refrains from the music objects.

In the rest of this paper, the number of the repeating pattern appearing in a feature string is called the *repeating frequency* of the repeating pattern.

3. The Correlative Matrix Approach for Finding Repeating Patterns

In this section we illustrate the basic idea of the proposed approach for finding repeating patterns through an example. The formal algorithms for finding repeating patterns are described in the Appendix.

We excerpt a phrase with 12 notes from Brahms Waltz in A flat, as shown in Figure 3. Its corresponding melody string **S** is "C6-Ab5-Ab5-C6-C6-Ab5-Ab5-C6-Db5-C6-Bb5-C6". There are three repeating patterns in this phrase, as shown in Table 1. Note that the proper sub-patterns of a repeating pattern are not included except they appear elsewhere in the phrase. For example, "C6-Ab5-Ab5-C6" is a repeating pattern and its repeating frequency is two. The repeating pattern "C6-Ab5-Ab5" is a proper substring of "C6-Ab5-Ab5-C6", whose repeating frequency is also two. Since "C6-Ab5-Ab5" appears only in the place where "C6-Ab5-Ab5-C6" appears, it is not listed in the table. However, if the sub-pattern also appears elsewhere in the phrase, it will be listed in the results. For example, "C6" is a repeating pattern whose repeating frequency is six, which means it appears more than "C6-Ab5-Ab5-C6" does in the phrase, therefore it is listed in the table.

Figure 3: A phrase excerpted from Brahms Waltz in A flat.

Repeating	Pattern Length	Repeating Pattern
2	4	C6-Ab5-Ab5-C6
6	1	C6
4	1	Ab5

Table 1: The repeating patterns in the phrase excerpted from Brahms Waltz in A flat.

To find all repeating patterns in the melody string **S**, an intuitive method is to generate all substrings of **S**. Then, each substring **X** of **S** will be compared with **S** to decide the number that **X** appears in **S**. This method is simple but very inefficient: if the length of string **S** is n, the number of all substrings of the input string **S** is $n+(n-1)+...+1$. For a substring **X** of length m, it needs $O(m \times n)$ character comparisons to decide the number that **X** appears in **S**. In the worst case, the total complexity will be $O(n^4)$. For example, for a music object consisting of 1000 notes, it will need $O(10^{12})$ character comparisons to find all repeating patterns in the music object.

To make the process of finding repeating patterns more efficient, we use a matrix, called *correlative matrix*, to keep the intermediate results of substring matching. For the i-th note and the j-th note in the melody string, if they are the same, the element in the i-th row and the j-th column of the correlative matrix will be set to one. Moreover, if the i-th note equals the j-th note and the $(i+1)$-th note equals the $(j+1)$-th note, which means there is a repeating pattern of length two, the element in the $(i+1)$-th row and the $(j+1)$-th column of the correlative matrix will be set to two. Therefore, the value of each element in the matrix denotes the length of a repeating pattern found. By computing the number of non-zero elements in the correlative matrix, the repeating frequencies of all repeating patterns can be found.

We use an example to illustrate the construction of the correlative matrix. For the example shown in Figure 3, since there are 12 notes in the melody string **S**, we initialize a 12×12 upper-triangular matrix, denoted by **T**, as shown in Table 2. We use $\mathbf{T}_{i,j}$ to indicate the element of the i-th row and j-th column in the matrix **T**.

	C6	Ab5	Ab5	C6	C6	Ab5	Ab5	C6	Db5	C6	Bb5	C6
C6	—											
Ab5		—										
Ab5			—									
C6				—								

C6					—						
Ab5						—					
Ab5							—				
C6								—			
Db5									—		
C6										—	
Bb5											—
C6											

Table 2: The initial correlative matrix for the music object in Figure 3.

The matrix will be filled row by row. For the first row, we compare the first note "C6" with each note in the melody string S. For each note S_j (denote the *j*-th note of S), if S_j = "C6" which means "C6" repeatedly appear in the first and *j*-th note in S, we set $T_{1,j} = 1$. In this example, since S_4, S_5, S_8, S_{10}, and S_{12} are "C6", $T_{1,4}$, $T_{1,5}$, $T_{1,8}$, $T_{1,10}$, and $T_{1,12}$ are set to one, as shown in Table 3.

	C6	Ab5	Ab5	C6	C6	Ab5	Ab5	C6	Db5	C6	Bb5	C6
C6	—			1	1			1		1		1
Ab5		—										
Ab5			—									
...												
...												
...												
C6												—

Table 3: The correlative matrix after the first note is processed.

Then we process the second note "Ab5". Since S_3 is also "Ab5", $T_{2,3}$ is set to one. For $T_{2,6}$, since S_6 = "Ab5"and $T_{1,5}$ is one, which implies the substring "C6-Ab5" repeatedly appears, $T_{2,6}$ is set to two. $T_{2,7}$ is set to one in the same way. The result is shown in Table 4.

	C6	Ab5	Ab5	C6	C6	Ab5	Ab5	C6	Db5	C6	Bb5	C6
C6	—			1	1			1		1		1
Ab5		—	1			2	1					
Ab5			—									
...												
...												
...												
C6												—

Table 4: The correlative matrix after the second note is processed.

For any two notes S_i and S_j ($i \neq j$ and $i, j > 1$) in the melody string S, if $S_i = S_j$, we set $T_{i,j} = T_{i-1,j-1} + 1$. If the value stored in $T_{i,j}$ is *n*, it indicates a repeating pattern of length *n* appearing in the positions $(j - n + 1)$ to j in S. Table 5 shows the result after all notes are processed.

After we construct the correlative matrix, the next step is to find all repeating patterns and their repeating frequencies. We use a set called the candidate set (denoted ***CS***) to record the repeating patterns and their repeating frequencies. Each element in the candidate set ***CS*** is of the form (*pattern*, *rep_count*, *sub_count*), where *pattern*, *rep_count*, and *sub_count* represent a repeating pattern, its repeating frequency, and the number of the repeating pattern being a substring of the other repeating patterns, respectively. The *sub_count* information is used to check whether this repeating pattern only appears as a substring of another repeating pattern (*i.e.*, *rep_count* = *sub_count*), in which case, it will not be listed in the final result.

	C6	Ab5	Ab5	C6	C6	Ab5	Ab5	C6	Db5	C6	Bb5	C6
C6	—			1	1			1		1		1
Ab5		—	1			2	1					
Ab5			—			1	3					
C6				—	1			4		1		1
C6					—			1		1		1
Ab5						—	1					
Ab5							—					
C6								—		1		1
Db5									—			
C6										—		1
Bb5											—	
C6												—

Table 5: The correlative matrix after all notes are processed.

The candidate set is initially set to an empty set. For every non-zero element $T_{i,j}$ in the correlative matrix **T**, its corresponding repeating patterns and repeating frequencies are computed and inserted into the candidate set ***CS***. According to the conditions (($T_{i,j} = 1$) or ($T_{i,j} > 1$))and (($T_{(i+1),(j+1)} = 0$) or ($T_{(i+1),(j+1)} \neq 0$)), there are four cases:

Case 1: ($T_{i,j} = 1$ and $T_{(i+1),(j+1)} = 0$). For example, the value of $T_{1,4}$ is one, which indicates the repeating frequency of the corresponding substring "C6" is one. Moreover, the value of $T_{2,5}$ is zero, which means "C6" is not a proper substring of another repeating pattern at this position. We insert ("C6", 1, 0) into the candidate set ***CS***.

Case 2: ($T_{i,j} = 1$ and $T_{(i+1),(j+1)} \neq 0$). For example, the value of $T_{1,5}$ is one, which indicates the repeating frequency of the corresponding substring "C6" is one. Moreover, the repeating frequency of $T_{2,6}$ is not zero, which means "C6" is a proper substring of another repeating pattern ("C6-Ab5") here. Since "C6" is already in the candidate set ***CS***, we modify ("C6", 1, 0) into ("C6", 2, 1) to record that another repetition of "C6" is found.

Case 3: ($T_{i,j} > 1$ and $T_{(i+1),(j+1)} \neq 0$). For example, the value of $T_{2,6}$ is two, which indicates the strings "C6-Ab5" and "Ab5" repeatedly appear here, we should insert ("C6-Ab5", 1, 0) and ("Ab5", 1, 0) into the candidate set ***CS***. However, since $T_{3,7}$ is three, which indicates the two repeating patterns "C6-Ab5" and "Ab5" are substrings of the repeating pattern "C6-Ab5-Ab5". Therefore, we insert ("C6-Ab5", 1, 1) and ("Ab5", 1, 1) into the candidate set ***CS*** instead.

Case 4: ($T_{i,j} > 1$ and $T_{(i+1),(j+1)} = 0$). For example, the value of $T_{4,8}$ is four, which indicates all of the four sub-strings "C6-Ab5-Ab5-C6", "Ab5-Ab5-C6", "Ab5-C6", and "C6", repeatedly appear here. Moreover, since $T_{5,9}$ is

zero, which indicates these repeating patterns are not a proper substring of another repeating pattern here, We insert ("C6-Ab5-Ab5-C6", 1, 0), ("Ab5-Ab5-C6", 1, 1), and ("Ab5-C6", 1, 1) into the candidate set ***CS*** and change ("C6", 6, 1) into ("C6", 7, 2).

After examining every non-zero element, the candidate set ***CS*** becomes {("C6", 15, 1), ("Ab5", 6, 2), ("C6-Ab5", 1, 1), ("C6-Ab5-Ab5", 1, 1), ("Ab5-Ab5", 1, 1), ("C6-Ab5-Ab5-C6", 1, 0), ("Ab5-Ab5-C6", 1, 1), and ("Ab5-C6", 1, 1)}. There are two more tasks we have to do. First, if a repeating pattern is a substring of another repeating pattern, and their repeating frequencies are the same, it will be removed from the candidate set ***CS***. In this example, the elements ("C6-Ab5", 1, 1), ("C6-Ab5-Ab5", 1, 1), ("Ab5-Ab5", 1, 1), ("Ab5-Ab5-C6", 1, 1), and ("Ab5-C6", 1, 1) are removed since they are all the substrings of the repeating pattern "C6-Ab5-Ab5-C6".

Second, we should calculate the real repeating frequency for every repeating pattern found. For the repeating pattern "C6", there are 15 non-zero elements associated with it in the correlative matrix. However, the repeating frequency of "C6" is six. It is because for a repeating pattern whose repeating frequency is f, there will be $C_2^f = \frac{f(f-1)}{2}$ nonzero elements associated with this repeating pattern. Therefore, for each element (*pattern*, *rep_count*, *sub_count*) in the candidate set ***CS***, the real repeating frequency of the repeating pattern *pattern* can be derived by the following equation:

$$rep\_count = \frac{f(f-1)}{2}, \; i.e., \; f = \frac{1+\sqrt{1+8\times rep\_count}}{2} \quad \text{(Eq. 1)}$$

For example, since the rep_count of the element ("C6", 15, 1) is 15, the repeating frequency f of the repeating pattern "C6" will be $f = \frac{1+\sqrt{1+8\times 15}}{2} = 6$. Similarly, the repeating frequencies of the repeating patterns "Ab5" and "C6-Ab5-Ab5-C6" are $\frac{1+\sqrt{1+8\times 6}}{2} = 4$ and $\frac{1+\sqrt{1+8\times 1}}{2} = 2$, respectively.

4. Experiments

To show the efficiency of the proposed algorithms, a series of experiments are performed. The results are further analyzed to show the effectiveness of our approach.

4.1 Experiment Set-up

We implement the repeating pattern finding algorithms by ANSI C and perform a series of experiments on a Sun Sparc20 workstation with SunOS 4.1.4. The performance is measured by elapsed time. There are two data sets used in these experiments. One contains synthetic music objects and the other contains real music objects. The synthetic music objects are generated with uniformly distributed notes. The real music objects are collected from websites, mainly the *Classical MIDI Archives Site* (http://www.prs.net). They are classical music objects of MIDI form composed by various composers. All music objects are parsed to extract their corresponding melody strings. The object size of a music object is measured by the length of the corresponding melody string. For the real music objects, their average object size is 619.3. The *note count* is defined as the number of distinct notes appearing in a melody string. According to the MIDI standard[18], the alphabet size is 128. Therefore, the note count of every melody string is between 1 and 128. According to the experiments, the average note count of real music objects is 23.2.

4.2 Performance Analysis

There are three factors dominating the performance of the proposed algorithms: the number of music objects, the object size of music objects, and the note counts of music objects. The first two experiments illustrate the elapsed time versus the number of music objects for the synthetic music data set and real music data set. For the synthetic music data, the object size and note count are set to 500 and 30, respectively. The elapsed time in both data set increases linearly as the number of music objects increases, as shown in Figure 4 and Figure 5. Moreover, the elapsed time of the real music data set is greater than that of the synthetic music data set. It is because in the synthetic music data set, the average length of the repeating patterns is much shorter than that in the real music data set. According to the experiments (Table 6 and Table 7), the average length of the longest repeating patterns of real music object is greater than ten while that of a synthetic music object is no greater than five.

Figure 6 and Figure 7 show the elapsed time versus the object size of music objects for the synthetic and real music data sets. In the synthetic case, the note count is set to 30. As Figure 6 and Figure 7 show, the elapsed time grows squarely as the size of the music objects increases for both data sets.

In the following, we show the effects of the note count on the performance in Figure 8 and Figure 9. In the real music data set, the note count of every real music object is less than 60. In both cases, the elapsed time decreases rapidly as the note count increases. It is because for music objects with small note counts, more repeating patterns are likely contained in them and the length of their longest repeating patterns tends to be longer. A more detailed analysis of the effect on the note count will be discussed in the next subsection.

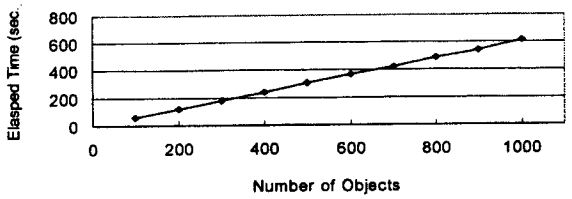


Figure 4: Elapsed time vs. number of objects for the synthetic music data set.

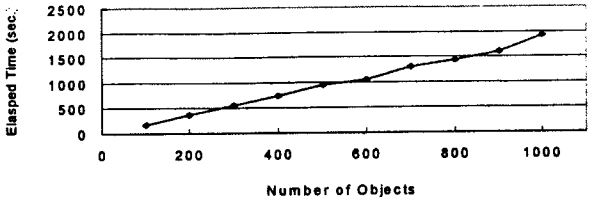


Figure 5: Elapsed time vs. number of objects for the real music data set.

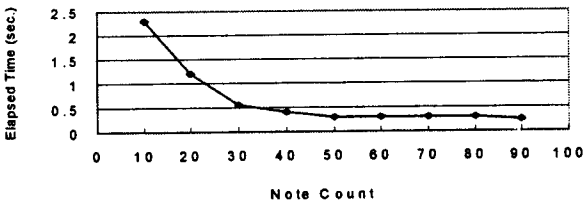


Figure 6: Elapsed time vs. object size for the synthetic music data set.

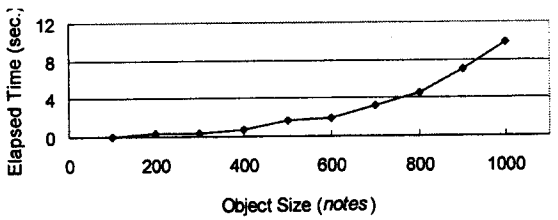


Figure 7: Elapsed time vs. object size for the real music data set.

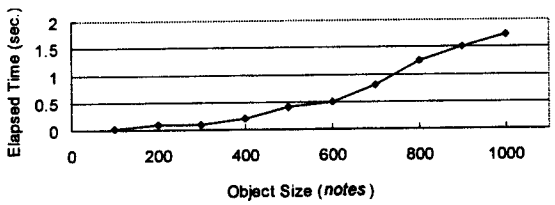


Figure 8: Elapsed time vs. note count for the synthetic music data set.

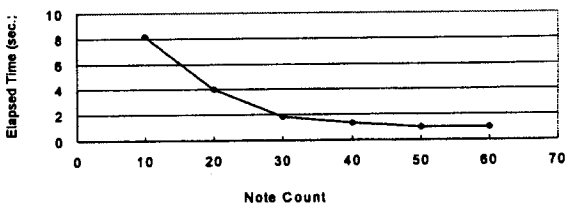


Figure 9: Elapsed time vs. note count for the real music data set.

Object ID	Composer	Opus.	Object Size	Number of Motives	Number of Repeating Patterns	Recall % Of Motives	the longest repeating pattern (freq., length)	Max. repeating frequency (freq., length)
Object_1	Beethoven	WoO.82	378	2	9	100 %	(2, 33)	(9, 3)
Object_2	J. S. Bach	BWV.772	252	1	14	100 %	(2, 7)	(4, 3)
Object_3	J. S. Bach	BWV.773	361	1	19	100 %	(2, 6)	(6, 3)
Object_4	J. S. Bach	BWV.774	276	1	10	100 %	(2, 12)	(4, 3)
Object_5	J. S. Bach	BWV.775	283	1	21	100 %	(2, 12)	(7, 3)
Object_6	J. S. Bach	BWV.776	389	2	16	100 %	(2, 7)	(3, 3)
Object_7	J. S. Bach	BWV.777	608	2	7	100 %	(2, 14)	(3, 3)
Object_8	Schubert	D.891	163	2	6	100 %	(2, 34)	(4, 3)
Object_9	Schubert	D.881	252	2	7	100 %	(2, 37)	(8, 6)
Object_10	Schubert	D.667	321	2	15	100 %	(2, 24)	(10, 3)

Table 7: Statistical information of testing data.

4.3 Semantics Analysis

In this subsection, we analyze the experiment results of both the real and synthetic music data sets. As we have mentioned above, for a real music object and synthetic music object of the same object size and note count, the length of the longest repeating pattern of the real music object is much longer than that of the synthetic music object. Table 6 shows the statistics for various lengths of the repeating patterns found in the synthetic music objects with 300 and 500 notes. We let the note counts to 20 and 30 since we found for most real music objects, their note counts are between 20 and 30. For the ten selected real music objects shown in Table 7, the length of their longest repeating patterns is each longer than six. However, the lengths of all repeating patterns found in Table 6 are shorter than five. This indicates many repeating patterns (at least

for the longest one) of a real music object are intentionally created by its composer.

Pattern Length	Object Size = 300 (*notes*)		Object Size = 500 (*notes*)	
	|A| = 20	|A| = 30	|A| = 20	|A| = 30
2	73	60	98	128
3	43	12	90	15
4	7	0	6	0
5	1	0	3	0
6	0	0	0	0
7	0	0	0	0
TOTAL	124	72	225	143

Table 6: The statistical information of discovered repeating patterns from the synthetic data.

Table 7 also shows that the motives of the ten real music objects are all extracted by the repeating pattern finding algorithms. We remove the repeating patterns whose lengths are shorter than or equal to two since they lack the semantics of repeating patterns. Also note that the shorter the length of a repeating pattern is, the larger the repeating frequency tends to be. The length of the longest repeating pattern in a music object also shows the characteristics of different composers. According to the preliminary experiments, the longest repeating pattern of J. S. Bach's "Two-Part Invention" is shorter than that of Beethoven's and Schubert's works. Further study on their complete works will be investigated.

The repeating patterns can be applied for music data clustering. For two repeating patterns, their editing distance, which is defined based on their longest common substring, is used as the dissimilarity measure of the two repeating patterns. Let **a** and **b** be two repeating patterns. The editing distance $d(\mathbf{a}, \mathbf{b})$, given by Wagner and Fischer, is defined as follows [23]:

$$d(\mathbf{a}, \mathbf{b}) = \text{length}(\mathbf{a}) + \text{length}(\mathbf{b}) - 2 \times q(\mathbf{a}, \mathbf{b}),$$

where length(**a**), length(**b**), and $q(\mathbf{a}, \mathbf{b})$ denote the length of repeating patterns **a** and **b**, and the longest common substring between **a** and **b**, respectively.

The music objects shown in Table 7 are clustered in two ways. Table 8 and Table 9 show the dissimilarity matrices of the ten music objects based on the editing distances of the melody strings and their longest repeating patterns, respectively.

Obj_ID	1	2	3	4	5	6	7	8	9	10
1	0	30	35	31	31	40	33	45	56	39
2	30	0	**11**	**15**	**17**	**14**	**13**	29	36	27
3	35	**11**	0	**16**	**14**	**11**	**16**	36	43	28
4	31	**15**	**16**	0	**20**	**19**	**14**	40	47	28
5	31	**17**	**14**	**20**	0	**19**	**24**	38	49	**24**
6	40	**14**	**11**	**19**	**19**	0	**21**	41	44	31
7	33	**13**	**16**	**14**	**24**	**21**	0	38	45	28
8	45	29	36	40	38	41	38	0	45	44
9	56	36	43	47	49	44	45	45	0	45
10	39	27	28	28	**24**	31	28	44	55	0

Table 8: The dissimilarity matrix for the longest repeating patterns of the music objects in Table 7.

The dissimilarity matrix is symmetric. The smaller value of an element represents a higher similarity between the two corresponding music objects. Note that the lengths of the melody strings and the longest repeating patterns of the two music objects may not be equal, thus the results are further normalized. As shown in the Table 8, the elements whose values are smaller than 25 are marked. Examining the Table 8, the dissimilarity values among Object_2, Object_3, Object_4, Object_5, Object_6, and Object_7 are less than 25. It suggests that these music objects be classified into the same cluster. Referring to Table 7, the composers of these music objects are also the same (J. S. Bach). On the other hand, the dissimilarity matrix (in Table 9) constructed from the melody strings does not classify Object_2, Object_3, Object_4, Object_5, Object_6, and Object_7 into the same cluster. Moreover, it suggests that Object_1 and Object_2 be classified into the same cluster. However, the two music objects are composed by different composers. This result shows the repeating pattern is a better music feature for clustering music objects.

Obj_ID	1	2	3	4	5	6	7	8	9	10
1	0	**96**	112	**98**	114	118	110	116	122	111
2	**96**	0	112	**98**	**82**	122	120	104	120	121
3	112	112	0	128	112	**96**	124	130	140	131
4	**98**	**98**	128	0	114	134	**96**	122	138	101
5	114	**82**	112	114	0	110	132	110	110	123
6	118	122	**96**	134	110	0	130	122	126	145
7	110	120	124	**96**	132	130	0	130	138	119
8	116	104	130	122	110	122	130	0	**90**	124
9	122	120	140	138	110	126	138	**90**	0	133
10	111	121	131	101	123	145	119	124	133	0

Table 9: The dissimilarity matrix for the melody strings of the music objects in Table 7.

5. Conclusion

Music is an important type of media. However, few works were focused on the music databases. Since repeating patterns can be found in both classical and pop music objects, we choose them as music features for content-based music data retrieval. In this paper, we propose an approach for extracting all repeating patterns in the melody string of a music object. By using the correlative matrix, we find repeating patterns based on their repeating substrings found. According to our experiments, for a music object with 1000 notes, we can find all its repeating patterns in ten seconds, which shows its feasibility for a practical use.

By analyzing the results, we find all motives are repeating patterns for the music objects sampled. We also find the repeating patterns of the music objects of a composer are more similar than those of other composers. Other musical semantics of the repeating patterns are currently being investigated. Moreover, issues on music data mining such as music data generalization, music data clustering, and music association rules are to be studied. Finally, since

some modifications can be done on a motive by the composer to make the music object sound more fruitful and variant, a repeating pattern may be approximately (not exactly) repeated. We will extend the proposed algorithms such that all approximate repeating patterns can also be found.

6. REFERENCES

[1] Abraham, G., ed., *The New Oxford History of Music, Vol. VIII and Vol. IX*, Oxford University Press, 1988.

[2] Bakhmutova, V., V. D. Gusev, and T. N. Titkova, "The Search for Adaptations in Song Melodies," *Computer Music Journal*, Vol. 21, No. 1, pp. 58-67, Spring 1997.

[3] Balaban, M., "The Music Structures Approach to Knowledge Representation for Music Processing," *Computer Music Journal*, Vol. 20, No. 2, pp. 96-111, Summer 1996.

[4] Berndt, D. J., and Clifford J., "Using Dynamic Time Warping to Find Patterns in Time Series," In *Proceedings of AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994, pp. 359-370.

[5] Chen, J. C. C. and A. L. P. Chen, "Query by Rhythm: An Approach for Song Retrieval in Music Databases," *In Proc. of IEEE Intl. Workshop on Research Issues in Data Engineering*, pp. 139-146, 1998.

[6] Chou, T. C., A. L. P. Chen, and C. C. Liu, "Music Databases: Indexing Techniques and Implementation," *in Proc. of IEEE Intl. Workshop on Multimedia Data Base Management System*, 1996.

[7] Davenport, G., T.A. Smith, and N. Pincever. "Cinematic Primitives for Multimedia," *IEEE Computer Graphics & Applications*, pages 67-74, July 1991.

[8] Day, Y. F., S. Pagtas, M. Iino, A. Khokhar, and A. Ghafoor, "Object-Oriented Conceptual Modeling of Video Data," *In Proc. of IEEE Data Engineering*, pages 401-408, 1995.

[9] Dowling, W. J., "Pitch Structure," in *Representing Musical Structure*, P. Howell, R. West, and I. Cross, ed., Academic Press, London, 1991.

[10] Ghias, A., Logan, H., Chamberlin, D., and Smith, B. C., "Query by Humming: Musical Information Retrieval in an Audio Database," In *Proceedings of Third ACM International Conference on Multimedia*, 1995, pp. 231-236.

[11] Hjelsvold, R. and R. Midtsraum, "Modeling and Querying Video Data," *In Int'l Conf. on Very Large Data Bases*, pages 686-694, 1994.

[12] Jones, G. T., *Music Theory*, Harper & Row, Publishers, New York, 1974.

[13] Krumhansl, C. L., *Cognitive Foundations of Musical Pitch*, Oxford University Press, New York, 1990.

[14] Leman, M. "Tone Context by Pattern Integration over Time," in *Readings in Computer-Generated Music*, D. Baggi , ed., IEEE Computer Society Press, Los Alamitos, 1992.

[15] Leman, M., "The Ontogenesis of Tone Semantics: Results of a Computer Study," in *Music and Connectionism*, P. Todd and G. Loy, ed., MIT Press, Cambridge, 1991.

[16] Li, J. Z. et al., "Modeling of video Spatial Relationships in an Object Database Management," *in Proc. of IEEE Workshop on Multimedia Database Management systems*, pp. 124-132, August 1996.

[17] Liu, C. C., A. J. L. Hsu, and A. L. P. Chen, "1D-List: An Approximate String Matching Algorithm for Content-Based Music Data Retrieval," submitted for publication.

[18] *General MIDI System Level 1 Specification*, MIDI Manufacturers Association (MMA) and the Japan MIDI Standards Committee (JMSC).

[19] Narmour, E., *The Analysis and Cognition of Basic Melodic Structures*, The University of Chicago Press, Chicago, 1990.

[20] Oomoto, E., and K. Tanaka, "OVID: Design and Implementation of a Video-Object Database System," *IEEE Transactions on Knowledge and Data Engineering*, pages 629-643, August 1993.

[21] Pfeiffer, S., Fischer, S., and Effelsberg, W., "Automatic Audio Content Analysis," In *Proceedings of the Fourth ACM International Multimedia Conference*, 1996, pp. 21-30

[22] Prather, R. E., "Harmonic Analysis from the Computer Representation of a Musical Score," *Communication of the ACM*, Vol. 39, No. 12, Dec. 1996, pp. 119, (pp. 239-255 of *Virtual Extension Edition of CACM*).

[23] Sankoff, D., and Kruskal J. B., eds., *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison*, Addison-Wesley Publishing Company, 1983.

[24] Smoliar, S. W. and HongJiang Zhang, "Content-Based Video Indexing and Retrieval," *IEEE Multimedia Magazine*, pages 62-72, 1994.

[25] Sundberg, J., A. Friberg, and L. Fryden, "Common Secrets of Musicians and Listeners: An Analysis-by-synthesis Study of Musical Performance," in *Representing Musical Structure*, P. Howell, R. West, and I. Cross, ed., Academic Press, London, 1991.

[26] Wold, E., Blum, T., Keislar, D., and Wheaton, J., "Content-based Classification, Search, and Retrieval of audio," *IEEE Multimedia*, Vol. 3, No. 3, Fall 1996, pp. 27-36.