Main Melody Extraction for Polyphonic Music

Hwei-Jen Lin, Hung-Hsuan Wu, and Yang-Ta Kao.

Department of Computer Sciences and Information Engineering

Tamkang University, Tamsui, Taipei, Taiwan, R.O.C.

E-mail：hjlin@cs.tku.edu.tw

Abstract

We propose an approach to main melody extraction from multi-track polyphonic MIDI files. In each track of a MIDI file, the system traces the pitch contour of the polyphonic music, and describes it in a monophonic form. In each of these contours, maximal repeating patterns are found using a correlative matrix. All of these patterns are then collected in a dictionary, with which we can find the set of all maximal repeating patterns with no redundancy, called the main melody. The main melody extraction results and how well these extracted main melodies can improve the work on content-based music retrieval are given and described.

Keywords：content-based retrieval, music information retrieval, repeating pattern, main melody extraction, correlative matrix, polyphonic music.

1. Introduction

The main melody is the set of certain patterns or phrases that the composer stresses and is always repeated throughout the entire piece of music [4][5]. The main melody or theme varies in nature, depending on the type of music. In the Baroque period, contrapuntal composition was very popular, a single idea, or theme, continued throughout the piece with scarcely a moment's letup [7]. Thus for main melody extraction, we are interested in finding the frequent occurring patterns in the music object.

Most of the published papers dealing with main melody extraction work on monophonic music [2] [4] [5]. In the recent years, there have been many papers presenting various methods of extracting main melodies in polyphonic music [3][1]. In this paper we propose a method for finding the maximal repeating patterns in not only monophonic but also polyphonic music stored in multi-track MIDI files. The MIDI file is pre-processed first to obtain the contour of each track. A correlative matrix is then utilized to determine all maximal repeating patterns in each track. A dictionary is then used to collect the patterns extracted from each track. Finally, some patterns are removed from the dictionary, such as redundant patterns, patterns that are proper sub-patterns of other patterns, and patterns that are part of the accompaniment.

This paper is organized as follows. In Section 2, the main melody extraction procedure is proposed. The experimental results are shown in Section 3. Section 4 concludes the paper and discusses some future work.

2. Main Melody Extraction

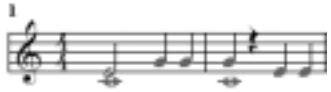
In this section, we propose a procedure of main melody extraction from polyphonic music. In each MIDI file track, the system traces out the pitch contour of the polyphonic music, which is described in a monophonic form. We construct a correlative matrix to find out all maximal repeating patterns in each track and use a dictionary to collect all repeating patterns with the numbers of their occurrences [5]. Finally, inappropriate candidates from the collection are discarded according to some general properties of the main melody. The set of remaining patterns in the dictionary is the so-called main melody.

2.1. Pitch Contour Tracing

The outer voice is usually perceptually significant, while the inner voice is hard to recognize [7]. In this research, the highest pitch contour will be traced for main melody extraction; that is, if there are more than two notes playing simultaneously, the highest pitch is extracted. This process is called the pitch contour tracing. Thus, the pitch contour of a piece of polyphonic music is the sequence of the highest pitches along the

music sequence.

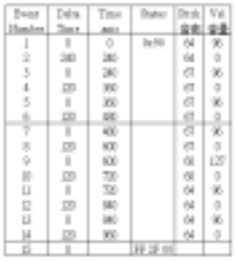
Every music note in one MIDI track is specified by two MIDI events, note-on and note-off. In a polyphonic MIDI file, the note-on and note-off events are interlaced in time sequence. See the example shown in Figures 1(a) and 1(b).



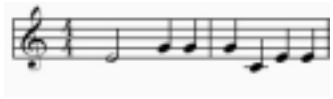
(a)

Event Number	Delta Time	Time axis	Status	Pitch	Vel
1	0	0	0x90	64	96
2	0	0	running	60	127
3	240	240	status	64	0
4	0	240		67	96
5	120	360		67	0
6	0	360		67	96
7	120	480		67	0
8	0	480		60	0
9	0	480		67	96
10	0	480		60	127
11	120	600		67	0
12	120	720		64	96
13	120	840		64	0
14	0	840		64	96
15	120	960		64	0
16	0	960		60	0
17	0		FF 2F 00		

(b)



(c)



(d)

Figure 1. (a). a piece of polyphonic music, (b). the MIDI events corresponding to (a), (c). the contour of (b) obtained by Algorithm PCTRACE, and (d). the corresponding music score of (c).

In this sub-section, we propose Algorithm PCTRACE to trace out the pitch contour for a piece of polyphonic music. For a given sequence of MIDI events along a time axis, PCTRACE utilizes a priority queue, denoted by PQ, to extract the highest pitch at every time point shown in the time axis. The sequence of these extracted highest pitches along the time axis is called the contour of the polyphonic music and denoted by CS in the Algorithm PCTRACE. The pitch contours of all tracks are extracted and then stored in a monophonic form. An example of the pitch contour tracing is shown in Figure 1.

Algorithm PCTRACE

The contour sequence *CS* is constructed as follows:

0. Initially, for time point 0, add a dummy pitch with value 0 into the empty sequences.
1. Along the time axis, all events at a time point *t* are read. For each note-off event, the system removes its corresponding note-on event from *PQ*. If this removed pitch is the largest value in *PQ*, the second largest pitch is extracted (not remove) and is added into *CS* for time point *t*.
2. Each note-on event is inserted into *PQ*.
3. The highest pitch, denoted by *hp*, is extracted from *PQ*, if *hp* is higher than the last pitch, denoted by *lp*, in *CS* then pitch *hp* is added into *CS* for time point *t*.
4. Proceed on the next time point along the time axis and go to step 1.

2.2. Maximal Repeating Pattern Extraction

Consider the example of the opening of the Brahms Waltz in A flat, shown in Figure 2. It can be represented by the pitch string C6–A^{b}5–A^{b}5–C6–C6–A^{b}5–A^{b}5–C6–D^{b}6–C6–B^{b}5–C6. We can easily see that the pattern C6–A^{b}5–A^{b}5–C6 occurs twice in the pitch string and is not a proper sub-pattern of any other repeating pattern. Such a pattern is a so-called maximal repeating pattern.

To automatically determine the maximal repeating patterns in a pitch string S, one may construct a correlative matrix T of size $n \times n$, where n is the length of the pitch string S. Let S_i denote the *i*-th character of S and $S_{i..j}$ denote the sub-string of S from the *i*-th to *j*-th characters. Initially, the value of each entry $T_{i,j}$ of T is set to zero's and then the correlative matrix is constructed row by row, from left to right as follows. If $S_i = S_j$, the value for $T_{i,j}$ is set to that of $T_{i-1,j-1}+1$; otherwise, set to zero. Because the matrix T is symmetric, we only need to work on the portion above the main diagonal of the matrix. The value of each entry $T_{i,j}$ denotes the length of the repeating pattern $S_{h..j}$, called the pattern corresponding to $T_{i,j}$, and thus, $j-h+1=T_{i,j}$ or $h=j+1-T_{i,j}$. The construction of the correlative matrix is performed by Algorithm CoMatrix given as follows.

Algorithm CoMatrix

//initialization
for $i \leftarrow 1$ to n do
for $j \leftarrow i+1$ to n do
$T_{i,j} = 0$
//construction of the correlative matrix
for $i \leftarrow 1$ to n do
for $j \leftarrow i+1$ to n do
if $(S_i = S_j)$ then $T_{i,j} \leftarrow T_{i-1,j-1}+1$

The correlative matrix for the Brahms Waltz opening shown in Figure 2 constructed by Algorithm CoMatrix is shown in Figure 3.

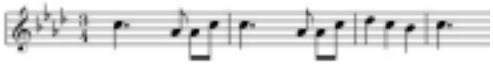


Figure 2. The score of the opening of the Brahms Waltz in A flat.

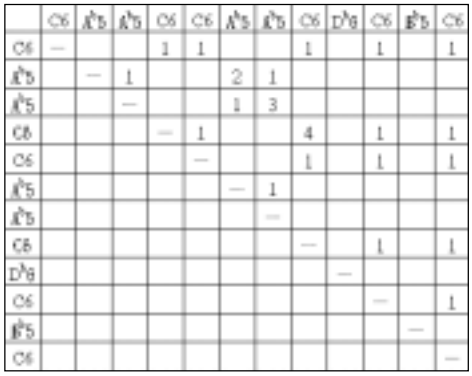


Figure 3. The correlative matrix of the opening pitch string of the Brahms Waltz in A flat.

For a repeating pattern that occurs *f* times in a pitch string, it must be counted for $n = C_2^f = f(f-1)/2$ times in the correlative matrix construction. Thus, $f = (1+\sqrt{1+8n})/2$.

We find that some short patterns tend to occur frequently. We simply discard these short repeating patterns by checking their lengths. That is, we discard a repeating pattern if its length is less than a given threshold value t.

2.3. Dictionary for Discarding Redundancy

There might be some repeating patterns that are proper sub-strings of other patterns. For the sake of efficiency, these proper sub-strings should be excluded in the main melody. That is, only the maximal repeating patterns instead of all repeating patterns are included. In order to detect only the maximal repeating patterns, Algorithm CoMatrix is modified as follows. In the case that $T_{i+1,j+1} = T_{i,j} + 1$, the pattern corresponding to $T_{i,j}$ is a proper sub-string of the pattern corresponding to $T_{i+1,j+1}$ and is not maximal. While in the case that $T_{i,j} \neq 0$ and $T_{i+1,j+1} = 0$, the pattern corresponding to $T_{i,j}$, or $S_{h..j}$, is maximal. The modified version of CoMatrix, named Modi-CoMatrix, is shown below. Algorithm CoMatrix utilizes a dictionary to save all repeating patterns extracted during the correlative matrix construction and record their occurrences.

Algorithm Modi-CoMatrix

//initialization
for $i \leftarrow 1$ to n do
for $j \leftarrow i+1$ to n do
$T_{i,j} = 0$
//construction of the correlative matrix
for $i \leftarrow 1$ to n do
for $j \leftarrow i+1$ to n do
if $(S_i = S_j)$ then $T_{i,j} \leftarrow T_{i-1,j-1}+1$
if $((T_{i,j} \geq t)$ and $(S_{i+1} \neq S_{j+1}))$ then
add the pattern corresponding to $T_{i,j}$
to the dictionary.

Compared to the previous method of correlative matrix construction, any repeating pattern that occurs f times in a pitch string of length k needs $C_2^f * k$ more comparisons and $C_2^f * (k-1)$ fewer times of adding repeating patterns into the dictionary. In programming aspect, the time taken by a comparison is much less than the time taken by a function call of inserting a pattern into a dictionary.

Whenever a maximal repeating pattern is detected during the correlative matrix construction, we check whether it exists in the dictionary or not. If it does, just increase the recorded number of the occurrences of that pattern by 1; otherwise, insert this pattern and initialize the number of its occurrences to 1.

In a MIDI file of multi-track music, there might be more than one track that contains some portion of the main melody. So we perform Algorithm Modi-CoMatrix on each track and

collect all extracted patterns in a dictionary. In order to obtain the optimal set of maximal repeating patterns, we need further remove proper sub-patterns using a string-matching technique on the resulting dictionary. All these patterns remaining in the dictionary are called the candidates of the main melody fragment, whose interval strings are stored for later use, where the interval string of a candidate is the sequence of differences between pairs of adjacent pitches in its pitch string.

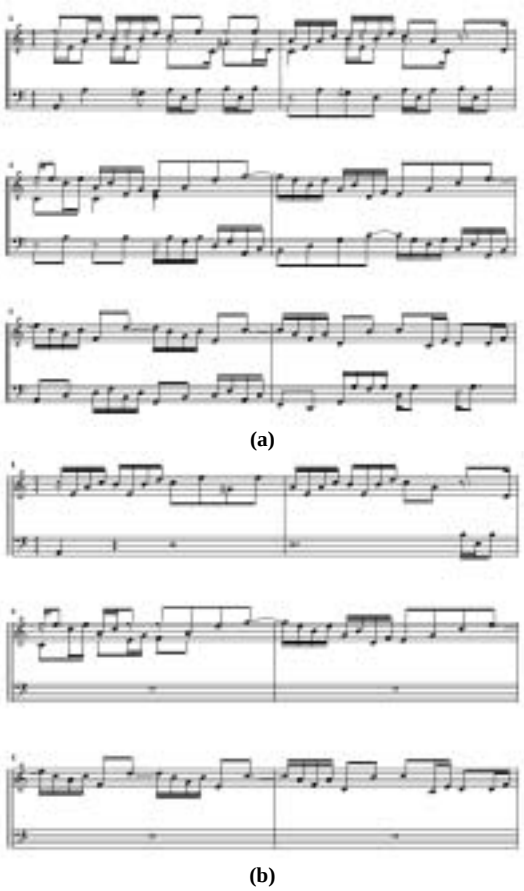
2.4. Removing Improper Candidates

Most melody lines contain a preponderance of conjunct motion, but the inclusion of few leaps or disjunction motion will help greatly in adding interest and variety to the melody line [6]. When extraction is performed on multi-track polyphonic music, the candidates might be extracted from the accompaniment part. The accompaniment might consist of many disjunctions, frequently repeating tones, or long scales, which are rarely included in the main melody. Thus, we ought to discard these candidates. According to the fact that the main melody consists of much more conjunction intervals than disjunction intervals. We may detect disjunctions in a candidate by checking its disjunction ratio R_{disj}, which is the ratio of the number of the disjunctions to the total number of intervals; that is, R_{disj} = *No. of disjunction intervals / No. of all intervals*. If the value of R_{disj} for a candidate is out of a given range, it has too many disjunctions or is lack of disjunctions, and thus, it possibly comes from the accompaniment and should be removed. Frequently repeating tones much more likely come from the accompaniment than the main melody. To detect frequently repeating tones occurring in a candidate, we check the zero interval ratio R_{zero}, which is the ratio of the number of the zero intervals to the total number of intervals; that is, R_{zero} = *No. of zero intervals / No. of all intervals*. If the valve of R_{zero} is greater than a given threshold value, we consider the candidate as part of the accompaniment and remove it. In this research we assume a disjunction interval is of more than 6 half tones (the perfect 4^{th}).

To detect the candidate patterns containing music scale, a window-based technique is applied as follows. If a candidate contains 7 consecutive intervals with the total length equal to 12, then it is identified to involve a music scale or a close music scale and is removed.

3. Experimental Results

No matter whether a MIDI file contains polyphonic or monophonic music, and the music is in multiple tracks or in a single track, Algorithm PCTRACE can trace out correct pitch contours. Figure 4(a) shows the opening 6 measures of the *Bach's 2-Invention No.13 in a minor* in a single-track MIDI file and its result of the pitch contour tracing is shown in Figure 4(b).



(a)

(b)

Figure 4. (a). The opening 6 measures of the Bach's 2-Invention No.13 and **(b).** The result of PCTRACE.

The time complexity of PCTRACE module is $O(n)$, where *n* is the number of notes in a MIDI file, which is approximately proportional to the size of the MIDI file. Figure 5 shows the time cost of PCTRACE versus the size of MIDI file.

In our experiment on removing the candidates coming from the accompaniment, the threshold value for R_{zero} is set to 0.66. If the value of R_{zero} for a candidate is greater than

0.66, we discard it. The given range for R_{disj} is from 0.05 to 0.5. If the value of R_{disj} for a candidate is out of the range, we discard it. To remove candidates involving a long scale, the threshold length of the scale for a candidate is set to half the number of notes in the candidate. We say that a candidate involves a long scale if it involves a scale occupying more than half portion of the whole string. If a candidate involves a long scale, it be removed.

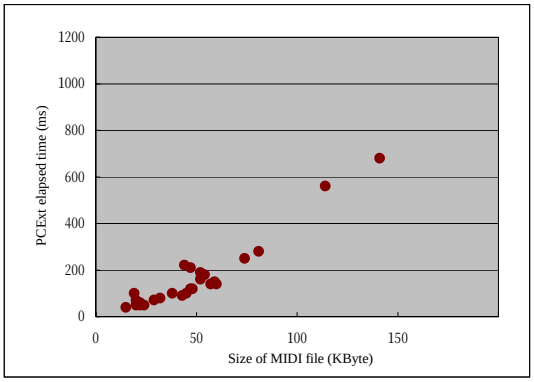


Figure 5. PCTRACE time cost v.s. size of MIDI file.

Figure 6 shows the time cost of the main melody extraction versus the number of notes. The work of main melody extraction includes constructing the correlative matrix, extracting all maximal repeating patterns and inserting them into in a dictionary, removing the proper sub-patterns, and discarding some improper candidates.

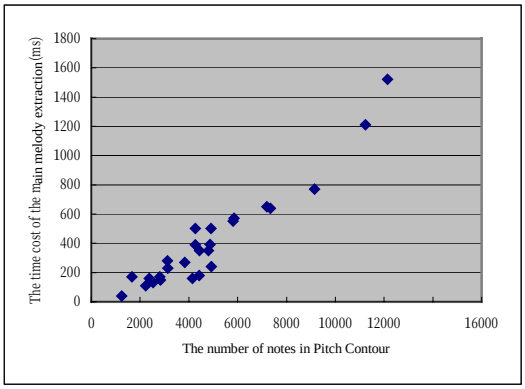


Figure 6. The time cost of the main melody extraction versus the number of notes in pitch contour.

From the experiments we found something interesting, the Baroque music usually have short and higher frequent occurrences candidates and the pop music usually have long candidates but lower frequent occurrences.

To overcome the modulate problem from queries, the interval strings, instead of the pitch contour strings of music are stored in the database. The main melody collection and the whole-song collection are saved in a database. For a query, at first the system searches in the main melody collection. If the search fails in the main melody collection, the system automatically searches for the query in the whole-song collection.

In the experiment, 30 songs are selected for query test from our music collection of 1135 songs. Some examples are shown in Figure 7. In these 30 queries, there are 26 queries hitting the main melody set and 4 queries hitting the music collection. About 86.66% of these queries hitting the main melody sets.

There are 1135 songs in our music collection. The average search time in main melody collection is about 215ms per song and the average search time in the whole-song collection is about 850ms. The average general search time in the database is about 300ms.

Name	Query fragment
Bach 2-Invention No.13 in a minor	
Piano Sonatian No. 14	
Mozart Piano Sonata No.14	
Mozart Symphony No.40, movement 1.	
Moudal	
Beethoven Symphony No. 9, movement 4	
Beethoven Symphony No.5 , movement 1	
Schubert Improment Op.90-2	
Schubert Improment Op.90-4	
All for One ： “I swear”	

Figure 7. Some examples of queries.

4. Conclusions and Future Work

In this paper, we propose an effective method for main melody extraction in multi-track polyphonic music. Correlative matrices are utilized to determine all maximal repeating patterns and a dictionary-based approach is applied to remove redundancy. According to some melody line properties inappropriate candidates are identified and removed.

We have defined main melody using the repeating property. However, there are several factors that can be taken into account such as the duration of notes and the phrases of the music.

If we can design a phrase tracking mechanism, we may treat a phrase as a pattern and use a dictionary to accumulate the number of occurrences of each phrase without using the correlative matrix. In improper MIDI file recording, it frequently occurs that the recording tempo is not matched with the performance tempo. This causes a time shift in the MIDI file. The time shift problem can be solved by evaluating all note durations and performing MIDI file quantization to obtain the correct performance tempo. If the correct tempo or phrase can be obtained, a dictionary-based approach can be employed to obtain the repeating patterns.

A partial repeating pattern problem can be managed. For example, if there are two patterns 'AB' and 'BC', where 'A', 'B' and 'C' are patterns longer than given a threshold, we create a virtual pattern 'ABC' to replace the two patterns. This approach can reduce the space for the main melody candidates and save the search time.

For repeating pattern search problem, many methods have been presented. Most of these methods have complexities in $O(n^2)$ with different constants [5][1], where n is the number of music notes. Some of them in $O(mn)$ [4], where m is the length of the longest repeating pattern and n is the length of the contour. We can produce further improvement by adopting this method.

Another open problem is the variation in the music work, which can be overcome by using one of the techniques for searching the longest common sequence, which are widely applied at gene engineering. The improvement of this system can also be achieved by the use of a voice transcription module to allow users to input a query by singing a piece of a song.

References：

[1] Colin Meek and William P. Birmingham, "*Thematic Extractor*", International Symposium on Music Information Retrieval, 2001, 119-128.

[2] H. Shih, Shrikanth S. Narayanan and C. C. Jay Kuo, "*Automatic Main Melody Extraction from MIDI Files with a Modified Lempel-Ziv Algorithm*". In Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing, Hong Kong, 2001.

[3] C. C. Liu, J. Hsu, Arbee L. P. Chen, "*Efficient Theme and Non-Trivial Repeating Pattern Discovering in Music Database*". In Proceedings IEEE International Conference on Data Engineering, 1999, 14-21,.

[4] Yuen-Hsien Tseng, "*Content-Based Retrieval for Music Collections*". In Proceedings of the ACM International Conference on Research and Development in Information Retrieval（SIGIR）, Berkeley, CA, 1999. 176-182.

[5] Jia-Lien Hsu, Arbee L. P. Chen, C. C. Liu, "*Efficient* Repeating *Pattern Finding in Music Database"*, Proceedings of the seventh international conference on Information and knowledge management, Bethesda, Maryland, United States, November 02-07, 1998, 281-288.

[6] Edward Aldwell and Carl Schachter, "*Harmony and Voice Leading, second edition*", Harcourt Brace Jovanovich College Publishers, 1989.

[7] Joseph Kerman, "*Listen*", Worth Publishers, 1976.