Geometric Measures of Distance between Two Pitch Contour Sequences

Hwei-Jen Lin[1,*] Hung-Hsuan Wu[1] Yang-Ta Kao[2]

[1] Department of Computer Science and Information Engineering

Tamkang University

Taipei, Taiwan, ROC

hjlin@cs.tku.edu.tw 123686@mail.tku.edu.tw

[2] Department of Information Network Technology

ChihLee Institute of Technology

Taipei, Taiwan, ROC

ydkao@mail.chihlee.edu.tw

Received: 3 March 2007; Revised: 8 July 2007 and 29 August 2007; Accepted: 14 September 2007

Abstract. Geometric measure is much more effective than others for melody matching, but is not fast enough to be practically used for on-line systems. In this paper, we propose an improved version of geometric music matching, in which the similarity measure for two melodies is defined as the area of the minimal region between them with the allowance of shifting one of the melodies in the horizontal and/or vertical directions. The matching efficiency is improved in two aspects. First, instead of absolute pitch, the pitch interval is used for matching to avoid the vertical shifting required in the search of the best matching. Second, the search time is further speeded up by using a branch-and-prune mechanism. The experimental results show that the time efficiency of the proposed geometric pattern matching is much more satisfactory than the other existing geometric matching methods for on-line music retrieval systems.

Keywords: content-based music retrieval, geometric matching, musical similarity, pitch interval, melody matching.

1 Introduction

With the rapid growth of digital audio data, content-based music retrieval has become a popular research topic in the past few years. Traditional music searching techniques done by text require some information such as the name of the song or the name of the composer. In many cases, the user might not remember such information, even though they might be able to hum or sing a fragment of the song. Thus, query by humming or singing would be more reasonable. Many different music retrieval systems have been created for academic research or business in recent years. Some of these systems allow users to search for music by inputting a fragment of the query music by singing or humming.

The algorithm for measuring music similarity in music matching is all important. Some critical factors must be considered including the quality of the query, feature selection, and the music data format. The first factor to be considered is that the input query might have various errors caused by user's imperfect memory, user's singing skill, or the quality of the pitch-tracking program [1]. A music information retrieval system should be able to tolerate such errors introduced by the users. To deal with this issue, an appropriate metric for musical similarity measure must be developed.

The second factor is feature selection. Music is made up of musical notes, each of which has many attributes such as pitch, duration, loudness, timbre, and so on. Byrd and Crawford [2] indicated that pitch alone is not sufficient for searching in a large database. L. A. Smith et al. [3] concluded that pitch and pitch duration are the most important attributes, and their findings will be followed in this research.

The third factor is the data format of the music. Symbolic music representation and audio wave format [4] are most commonly used in practice. Symbolic music representation specifies many musical attributes of music notes including pitch, duration, and loudness. The most commonly used symbolic music representation is the MIDI file. The audio wave format specifies the amplitude of sound over a time series. In this research, we adopt the symbolic music representation, which clearly specifies the information regarding pitch and duration.

* Correspondence author

Techniques based on string matching have been extensively explored in the literature, including edit distance [5] and n-gram [6,7,8]. These techniques can provide approximate matching. However, the features utilized in these techniques are all pitch sequences which have the drawback of retrieving too many irrelevant music documents with similar pitch but with a wide range of rhythm information.

Roger Jang et al. [9] proposed a "Dynamic Time-Warping" technique to search similar music documents in their collection of popular Chinese songs for a given query. In this technique, pitch-duration information is used as feature and dynamic programming is employed to search for the songs best matching the query. Their system tolerates different keys and variant speed between the query and matching songs. The main drawback of the time-warping method is its low efficiency due to the sampling of the audio query in a small time period. In order to reduce the search time, their system performed the search in multiple servers in parallel, or allows the users to choose a simpler search mechanism that matches the query against only the beginning of the reference song. However, this method might miss some possible matches.

Wiering et al. [10] proposed the representation of music notation as a weighted dot pattern, defined the Earth Mover's Distance for distance measure, and then used linear programming to compute the best match. The main drawback of this system is its high time-consumption.

Clifford et al. [11] considered the music notes in a melody as a set of points in a two-dimensional Euclidean space, and found the maximal partial subset match of two point sets by hashing two point sets into a table of size $O(n)$ and then applying binary wildcard matching to find the offset of the best match. However, this approach is an approximate geometric matching for possible collisions and its discrete point representation has the drawback of low tolerance to slight displacement of points.

Zhu et al. [12] considered each music note as a horizontal line segment. The slope of each line joining the starting point of a local minimum and the starting point of its nearest local maximum was evaluated. The hummed melody was matched against the reference melody in the horizontal direction according to the slope similarity. In each horizontal alignment, the hummed melody was moved in the vertical direction to a position so that the two compared melodies have the same mean pitch, and then the area between two melody contours were computed and defined as their dissimilarity. This approach has two drawbacks: it is very sensitive to noise and it tends to cause errors due to the alignment of two melodies in vertical direction according to their mean pitch.

D. Ó. Maidín [13] first used the geometric matching technique for music retrieval by considering both the pitch and rhythm information and representing music notes in a two-dimensional pitch-time space. Each note is represented as a horizontal line segment so that a sequence of notes can be described as a rectangular contour. This technique has two drawbacks. First, its representation of a note as a multiple of eighth notes causes some computational error. The second drawback is its assumption that two compared songs have the same tempo, which disables matching invariant to tempo.

Francu and Nevill-Manning [1] followed the geometric matching technique for computing the distance between a query and a reference. They transposed the key and rescaled the tempo of the query to search for the minimal average difference of the pitches of two melodies. In order to achieve this task, the contour is sampled and quantized into a sequence of notes of equal duration (20 milliseconds). Fixing the query in the horizontal direction, the median of the differences in pitch between the two melodies is determined, and the query is then transposed by this median difference so that the area is minimized in the horizontal direction. The query is then shifted note by note in the horizontal direction to align with the reference, and the minimal area in each direction can then be evaluated in $O(n)$ time. The distance between the two melodies is then defined as the minimal of these minimal areas and needs $O(mn)$ time to evaluate, where m and n are the numbers of the (sampled) notes in the query and the reference, respectively. In order to allow different tempos between query and reference, they suggested rescaling the query in different tempos with factors varying between 0.5 and 2 in coarse steps of 0.1 to find the best match. Generally speaking, most notes last hundreds of milliseconds, and thus quantization of a contour into a sequence of durations of 20 milliseconds increases the time cost for matching.

Aloupis et al. [14] generalized the geometric matching problem by releasing the limit of the note duration and by providing proof for the time-complexity. Given a fixed reference melody, the query melody is shifted vertically and horizontally in order to find the minimal area between them. They indicated that the minimal area must occur when two vertical edges coincide, and they proposed a binary tree search for the best vertical shifting step to further reduce the search time in the vertical direction. The overall search time then becomes $O(mn \log n)$.

Lubiw and Tanur [15] also used geometric matching to deal with the music retrieval problem on polyphonic music. The time complexity of their method is the same as that of Aloupis et al. with an additional large hidden constant coefficient. This coefficient necessary for tackling polyphonic music geometric matching is much greater than the size of a query. However, their method showed how to detect all occurrences of a query only in a single polyphonic reference melody rather than in a large music collection. Although their search result is satisfactory, the time efficiency is not satisfactory due to the large hidden coefficient.

Geometric matching based on search of minimal area between two melodies originally proposed by D. Ó. Maidín [13] and improved by Aloupis et al. [14] seems to be attractive due to its effectiveness. However, D. Ó. Maidín only juxtaposed two melodies of equal duration and transposed one melody in the vertical direction to evaluate the minimum area between them. Aloupis et al. generalized this approach and theoretically analyzed its

time complexity. Although its time complexity looks admissible due to the usage of a binary tree that supports the search in the vertical direction, the time cost in practice is not low due to the necessity of tree maintenance. This motivated us to improve the time efficiency of the naïve geometric matching method proposed by D. Ó. Maidín rather than proposed by Aloupis et al.. We improved the matching efficiency in two aspects: Using pitch interval instead of absolute pitch for matching to avoid the vertical shifting required in the search of the best matching, and employing a branch-and-prune mechanism to speed up the search in the horizontal direction.

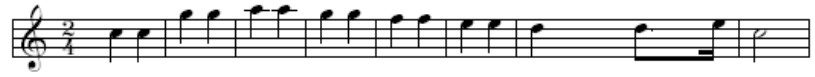
The rest of this paper is organized as follows. Section 2 provides a detailed description of our proposed method. Section 3 shows the experimental results of the proposed method and compares them with some other methods. Finally, in Section 4 we draw our conclusions and provide suggestions for future work.

2 Geometrical Matching

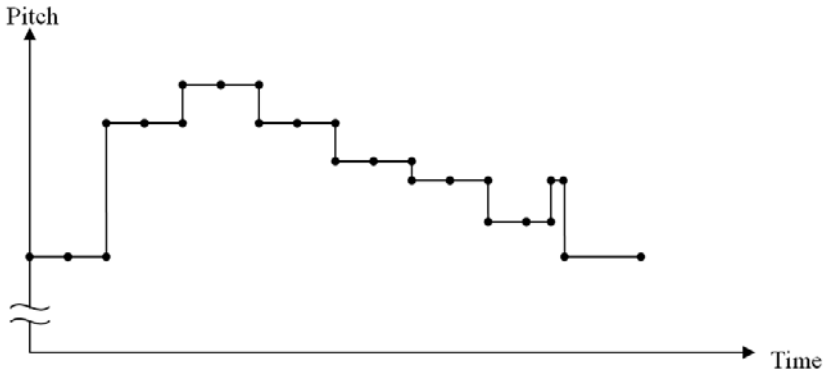
The aim of a music retrieval system is to match a short query melody against a larger reference melody and establish and rank the relevant references according to the similarity measurement. In Section 2.1, we briefly introduce the geometric matching technique proposed by Aloupis et al. [14]. We then describe how to compute the pitch interval area between a query and a reference in Section 2.2, and in Section 2.3, we show how to employ a branch-and-prune mechanism to enhance the matching speed for both the naïve geometric matching and our pitch interval geometric matching.

2.1 Pitch-Duration Geometric Matching

Aloupis et al. represented a melody as a sequence of music notes. Each note was described by a horizontal line segment, of which the height and the width denote the pitch value and the duration of the note, respectively. Such a sequence is called a pitch-duration sequence. For any two adjacent notes in different pitches, we connect the end point of the former and the starting point of the latter by a vertical edge. In such a way, a melody can be expressed as an orthogonal chain. Fig. 1(a) and Fig. 1(b) show a fragment of Mozart's Variations on 'Ah, vous dirai–je, maman', K. 265 and its corresponding orthogonal chain, respectively.



(a)



(b)

Fig. 1. (a). A fragment of Mozart's Variations on 'Ah, vous dirai–je, maman', K. 265, (b). Orthogonal chain representation of (a)

For a query $Q = (q_1, q_2, \ldots, q_m)$ and a reference $R = (r_1, r_2, \ldots, r_n)$, represented by sequences of notes, where q_i and r_j are the ith note of the query and the jth note of the reference, respectively. Let $P(q)$ denote the pitch value of a note q for later use. Aloupis et al. [14] shifted the query horizontally and vertically to find the minimal area between the query and the reference, and defined this minimal area as their distance. The region between the reference and a shifted version of the query is partitioned into rectangles C_α, $\alpha = 1, 2, \ldots, k$, by the extended vertical edges of both melodies, as shown in Fig. 2. Each rectangle is determined by two vertical lines and two horizontal edges. The height of each rectangle is the pitch difference of two notes in the two melodies. Therefore,

the area can be expressed as $A = \sum_{\alpha=1}^{k} |C_\alpha| = \sum_{\alpha=1}^{k} w_\alpha |P[q_{i_\alpha}] - P[r_{j_\alpha}] - \Delta|$, where $|C_\alpha|$ is the area of C_α, $P[q_{i_\alpha}]$ and $P[r_{j_\alpha}]$ denote the respect pitches in the query and the reference forming the αth rectangle C_α, and Δ denotes a vertical shifting step of the query. The area function can be minimized by setting the value of Δ to the weighted median of the differences $d_\alpha = P[q_{i_\alpha}] - P[r_{j_\alpha}]$. The weighted median can be found in $O(k)$ time. Usually $k = O(m+n)$, but $k = O(m)$ on the average.

Considering the horizontal movement of the query, Aloupis et al. indicated that the minimal area must occur in a case when two vertical edges coincide, one from the query and the other from the reference. There are at most $O(mn)$ (since there might be a case with more than one coincident pairs) such cases, and thus the query have to be shifted at most mn times and evaluate the area between the reference and each of the shifted versions of the query. Thus the minimal area between two melodies can be computed in $O(mn^2)$ time. They also proposed a binary tree to further speed up the search of the best vertical position and ease area reevaluation at each position. The computational time is $O(mn(m + \log n))$ rather than $O(mn \log n)$ as they claimed. To allow the efficient reevaluation of the area after a horizontal shift, the shift step must be the minimum of all possible steps that cause another pair of vertical lines coincide. Determination of such a step requires at least $\Theta(m)$ time, the search of the best vertical position requires $O(\log n)$ time, and tree modification (deletion and insertion) and reevaluation together require $O(\log n)$ time. Thus the total time required is $O(mn(m + \log n))$. The computational time is $O(mn(m + \log m)) = O(m^2 n)$, for $k = O(m)$ on the average.

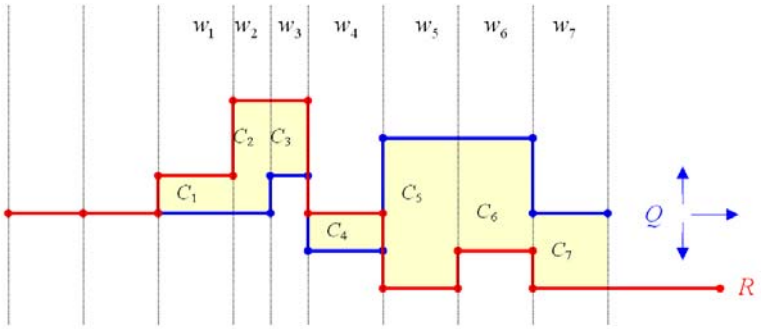


Fig. 2. The region between two melodies is partitioned into 7 rectangles

2.2 Pitch Interval-Duration Geometric Matching

Due to the fact that two aurally similar melodies have similar outlines, which are key-invariant, we adopt a pitch interval-duration sequence (called PID sequence), instead of a pitch-duration sequence, to characterizes and represent a melody. The pitch interval of a note is the difference of the pitch of the note from that of the preceding one. With the representation of PID sequence there is no need to shift the query in the vertical direction during the search of the minimal area, and so that the computational time can be dramatically reduced. Before transferring a given melody into a pitch interval sequence, adjacent notes of the same pitch in the melody are merged into one. In this paper, we measure similarity between two melodies based on their sequences of pitch intervals with duration, and call the matching mechanism based on this measurement the pitch interval geometric matching.

Each note in a sequence of pitch intervals can be also described by a horizontal edge in a two-dimensional space, of which the vertical axis corresponds to pitch interval value and the horizontal axis corresponds to time. For a query Q and a reference R, let $PI\_Q[i] = P[q_i] - P[q_{i-1}]$ denote the ith pitch interval, $i = 2, 3, \ldots, m$, and especially let $PI\_Q[1] = 0$. If we represent the ith pitch interval of Q as $< start\_Q[i], PI\_Q[i] >$, where $start\_Q[i]$ is the beginning time of the ith note q_i, then the PID sequence of Q can be represented as $(< start\_Q[i], PI\_Q[i] >)_{i = 1, 2, \ldots, m}$. With the assumption that the ending time of one note is the same as the beginning time of the next note, there is no need to record the ending time of any note, except the last note, for which we just add a dummy note in the end of the sequence to record the ending time of the last note. The pitch interval representation for the reference R is defined in the same fashion, and the jth pitch interval of R is represented by $< start\_R[j], PI\_R[j] >$, and the PID sequence of R is represented by $(< start\_R[j], PI\_R[j] >)_{j = 1, 2, \ldots, n}$. Note that the start time of the first note in each of the melodies is set to zero; that is, $start\_Q[1] = start\_R[1] = 0$.

With the use of the pitch interval instead of absolute pitch, it is no need to shift the query in the vertical direction. Without loss of generality, we assume the reference sequence is fixed. So we need only to shift the query sequence horizontally to find out the best alignment (minimal area). As depicted in Fig. 3, the query and the

reference are modeled as monotonic pitch interval rectilinear functions of time. Like the method proposed by Aloupis et al., the region between two melodies is partitioned into rectangles C_α, $\alpha = 1, 2, \ldots, k$, each defined by two vertical edges and two horizontal segments, where each vertical edge occurs at an end point of a note and each horizontal segment corresponds to a pitch interval. We evaluate the pitch interval area as $A = \sum_{\alpha=1}^{k}|C_\alpha| = \sum_{\alpha=1}^{k} w_\alpha \, |PI\_Q[i_\alpha] - PI\_R[j_\alpha]|$, where w_α and $|PI\_Q[i_\alpha] - PI\_R[j_\alpha]|$ are the width and the height of the αth rectangle C_α formed by pitch intervals $PI\_Q[i_\alpha]$ and $PI\_R[j_\alpha]$. For each shift of the query in the horizontal direction, we set $|C_\alpha| = 0$ if $i_\alpha = 1$ or $j_\alpha = 1$, since the region formed by $PI\_Q[1]$ or $PI\_R[1]$ is invalid and its area should be ignored. Each pitch interval area is evaluated in Usually $k = O(m+n)$, but again $k = O(m)$ on the average.

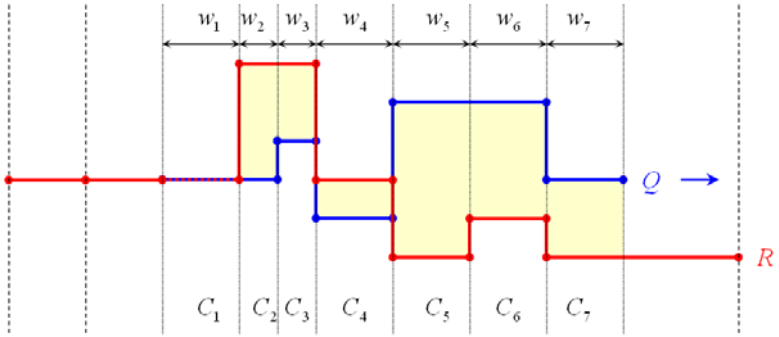


Fig. 3. The PID sequence of query Q is moved from left to right to search for the minimal pitch interval area

Since the minimal area between two melodies must occur at one of the cases when two vertical edges of the two melodies coincide and there might be some duplication over such cases (i.e. more than two coinciding edge pairs occur at the same time), there are at most mn possible horizontal positions needed to be evaluated. Therefore, there are at most $O(mn)$ different regions to be evaluated for area, and thus it takes $O(m^2n)$ time to find the minimal area. Rigorously speaking, the running times for our algorithm and the naïve geometric matching algorithm proposed by D. Ó. Maidín are $4m^2n+O(mn)$ and $10.5m^2n+O(mn)$, respectively. Both algorithms are in $O(m^2n)$ but with different leading constants, *4* and *10.5*. D. Ó. Maidín's algorithm requires a larger leading constant due to the necessity for the query to shift in the vertical direction to find the best match.

Due to the fact that the minimal area between two melodies must occur at a case when two vertical edges of the two melodies coincide, the minimal area can be determined in an efficient way described in the following. The query is shifted from left to right, starting from the beginning of the reference. Each shift is such that two vertical edges respectively from the two melodies coincide. As described in the following algorithm ***Pitch-Interval-Area*** for evaluating the minimal area, to determine the step size of next shift we evaluate the distance from the start time of each note q_i to the nearest of the following start times of the reference notes, denoted by *step*[*i*], for $i = 1, 2, \ldots, m$, and store them into a priority queue (a min heap) *H*. The step size of next shift, *shift-step*, is the minimal value in the heap *H,* i.e., the minimum of *step*[*i*]'s. Moreover, the index array *after*[*i*] = *j* represents the *j*th note of reference PID sequence nearest to the *i*th note of the query PID sequence. The three sub-steps in the while-loop respectively update the area, evaluate *shifted-step*, and update the starting points of all notes in the query PID. In such a way all cases that two vertical edges respectively from the two melodies coincide must be met, and thus the minimal area can be found.

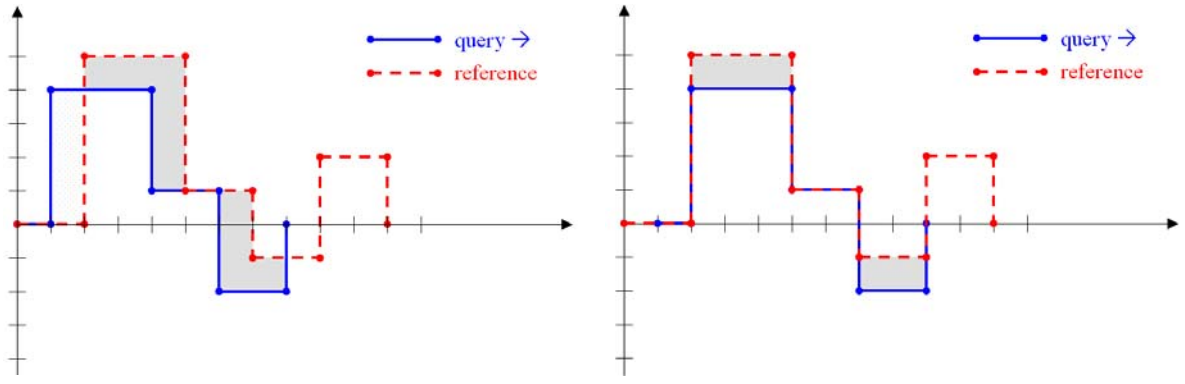
Algorithm *Pitch-Interval-Area*

1. Input: a query PID sequence $Q = \langle start\_Q[i], PI\_Q[i] \rangle_{i = 1, 2, \ldots, m}$ and
   a reference PID sequence $R = \langle start\_R[j], PI\_R[j] \rangle_{j = 1, 2, \ldots, n}$
   Output: MinArea
2. //Initialize the shifted query $Q' = \langle start\_Q'[i], PI\_Q[i] \rangle_{i = 1, 2, \ldots, m}$ and *after*[*i*] as the index of the reference note whose start time is behind and closest to the start time of the query note q_i
   MinArea ← ∞

```
    For i ← 1 to m do
        start_Q'[i] ← start_Q[i]
        PI_Q'[i] ← PI_Q[i]
        after[i] ← argmin{start_R[j]|start_R[j]>start_Q'[i]}
                   1≤j≤n
        step[i] ← start_R[after[i]] − start_Q'[i]
3.  While ( after[m] ≦n ) do
        A ←∑_{α=1}^{k} w_α | PI_Q'[i_α]−PI_R[j_α] |
        If A < MinArea then MinArea ← A
        shift-step ← min step[i]
                     1≤i≤m
        For i ← 1 to m do
            start_Q'[i] ← start_Q'[i] + shift-step
            If step[i] = shift-step
                then after[i] ← after[i] + 1
                     step[i] ← start_R[after[i]] − start_Q'[i]
                else step[i] ← step[i] − shift-step
```

As shown in Fig. 4(a), the PID sequences of a query and a reference: $(< start\_Q[i], PI\_Q[i] >)_{i = 1, 2, \ldots, 5}$ = (<0, 0>, <1, +4>, <4, +1>, <6, –2>, <8, 0>) and $(< start\_R[j], PI\_R[j] >)_{j = 1, 2, \ldots, 6}$ = (<0, 0>, <2, +5>, <5, +1>, <7, –1>, <9, 2>, <11, 0>), respectively, with the reference (< *start_R*[*j*], *PI_ R*[*j*] >) fixed, the query (< *start_Q*[*i*], *PI_Q*[*i*] >) is shifted from left to right. For the first note of PID sequence of the query, the nearest note of reference is 2; that is, *after*[1] = 2, from which we obtain *step*[1] = *start_R*[2] – *start_Q*[1] = 2. In the same fashion, we can obtain the following: *after*[2] = 2, *after*[3] = 3, *after*[4] = 4, *after*[5]=5, and *step*[2] = 1, *step*[3] = 1, *step*[4] = 1, *step*[5] = 1.

The while-loop evaluates the area and *shift-step*, and then shifts the query. The region between the two melodies is partitioned into 7 rectangles. As depicted in Fig. 4(a), the area of first and second rectangles cannot be taken account since the region formed by *PI_Q*[1] or *PI_R*[1] is invalid and its area should be ignored. Thus the area of the region between the two melodies is equal to 10, and the *shift-step* is equal to 1. After a shift of step size 1, the shifted query becomes $(< start\_Q'[i], PI\_Q'[i] >)_{i = 1, 2, \ldots, 5}$ = (<1, 0>, <2, +4>, <5, +1>, <7, –2>, <9, 0>), as depicted in Fig. 4(b). Since *step*[*i*] = *shift-step* for *i* = 2, 3, 4, and 5, the four corresponding rectangles are to be vanished, and value of each *after*[*i*] is incremented by 1 so that *after*[2] = 3, *after*[3] = 4, *after*[4] = 5, *after*[5] = 6. The value *after*[1] = 2 remains unchanged. The area between the two melodies then becomes 5 after this shift. The procedure is repeated until *after*[*m*] > *n*. Through the procedure the minimal area MiniArea can be found.



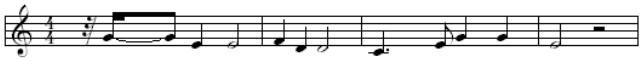
(a) (b)

Fig. 4. (a). The initial state of two PID sequences with *step*[1] = 2, *step*[2] = *step*[3] = *step*[4] = *step*[5] = 1, *shift-step* = 1, (b). The query in (a) is shifted by 1 unit

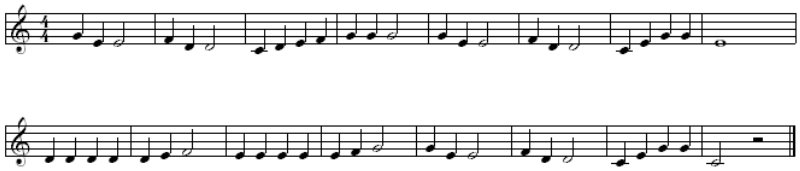
2.3 Branch and Prune

In the geometric matching process, the query is shifted and aligned against the fixed reference from left (beginning) to right (end). The pitch interval area A(δ) is a continuous function of the shifting step δ. Fig. 5(c) shows

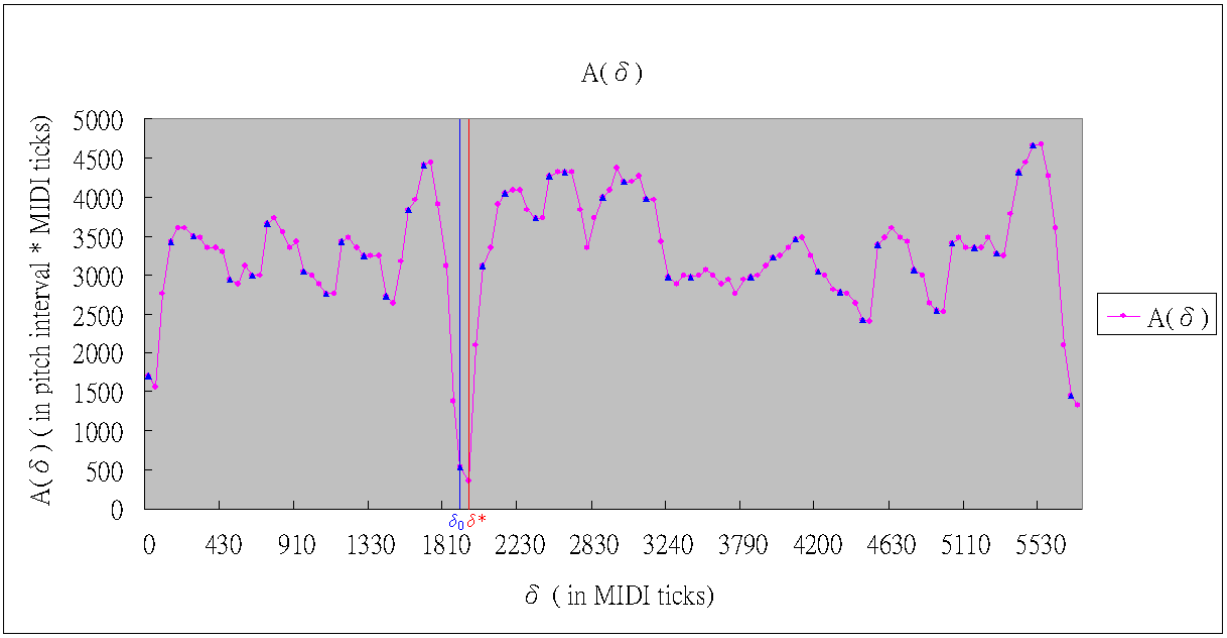
the sketch of A(δ) for the query melody given in Fig. 5(a) against the reference melody given in Fig. 5(b). The horizontal and the vertical axes respectively correspond to the shifting step δ and the pitch interval area A(δ). The small triangular points in Fig. 5(c) indicate the shifting step values causing the start time of the first note of the query and the start time of some note in the reference to be aligned; that is, *start_Q'*[1] = *start_Q*[1] + δ = *start_R*[*j*] for some *j*. We call the shifting steps indicated by these vertical lines the steps of coarse alignment. Observing from our preliminary experiments, around the global minimum there is at least a point of coarse alignment whose corresponding area is rather small. From the sketch of A(δ) shown in Fig. 5(c), it can be seen that the global minimum of A(δ) occurs at $\delta = \delta^* = 1930$ MIDI ticks; while the coarse alignment at $\delta = \delta_0 = 1920$ MIDI ticks nearby has a small value of area A(δ_0). This observation motivated us to further improve the matching by employing a branch-and-prune mechanism.



(a)



(b)



(c)

Fig. 5. (a). Query of a folk song, (b). A reference to be matched, (c). The sketch of A(δ) for the query against the reference

First, we consider only the cases that the start time of the first note of the shifted version of the query, *start_Q'*[*i*], is aligned against the start time of a note, *start_R*[*j*], in the reference, and for each case causing a small area (in this paper we consider the cases causing top 5% smallest areas) we then shift the query within only the interval around the point of the alignment to search for a minimal area.

Initially, the shifted version of the query, *start_Q'*[*i*], is set to the input query *start_Q*[*i*] in step 2, and then in each subsequent *j*th step of coarse alignment, *start_Q'*[*i*] is modified by adding the total shifting step *start_R*[*j*] – *start_Q*[1] for each *i*, and the area *A*[*j*] is reevaluated in step 3. There are at most *n* cases in which *start_Q'*[1] is aligned against *start_R*[*j*] and for each position the area can be evaluated in *O*(*m*) time, thus the time complexity

at this stage is at most $O(mn)$. In the following stage, search for the approximate area around each of the positions causing top 5% of the smallest areas. For instance, for a selected position *start_R*[*j*] we align *start_Q'*[1] around the interval [*start_R*[*j*–1], *start_R*[*j*+1]]. Since *start_Q'*[1] is aligned against the reference over only about 10% (2×5%) of the PID of the reference, the second stage costs about 10% of the time required by the method without the aid of the branch-and-prune mechanism. Although the search yields a local minimum, most of these local minima is either close to the global one or equals the global one. This is due to the fact that for most of the best matches the first note of the query is exactly aligned with a note in the reference. Therefore, approximation yielded by the branch-and-prune mechanism would not evidently degrade the matching accuracy. This improved algorithm ***B&P Pitch Interval Area*** is given in the following.

Algorithm *B&P-Pitch-Interval-Area*

```
1. Input: the query PID sequence Q = <start_Q([i], PI_Q[i]>_{i = 1, 2, …, m} and
          the reference PID sequence R = <start_R[j], PI_R[j]>_{j = 1, 2, …, n}
   Output: MinArea
2. //Initialize the shifted query version Q'
   For i ← 1 to m do
       start_Q'[i] ← start_Q[i]
       PI_Q'[i] ← PI_Q[i]
3. //Compute the area obtained from each of the alignment that the first
   note of the query matched with a note of the reference
   For j ← 1 to n−m+1 do
       For i ← 1 to m do
           start_Q'[i] ← start_Q[i] + (start_R[j] − start_Q[1])
       A[j] ← Σ_{α=1}^{k} w_α | PI_Q'[i_α]−PI_R[j_α] |
4. //Find positions of query causing top 5% minimal areas
   Set ←{1≦j≦n−m+1|A[j] is in top 5% minimal areas}
5. //Evaluate minimal area around each of the top 5% minimal areas
   MinArea ← ∞
   For each j in Set
       For i ← 1 to m do
           start_Q'[i] ← start_Q[i] + (start_R[j-1] − start_Q[1])
           after[i] ← argmin_{1≤j≤n} {start_R[j]|start_R[j]>start_Q'[i]}
           step[i] ← start_R[after[i]] − start_Q'[i]
       While ((after[1] ≦start_R[j+2] ) and (after[m] ≦n))
           // Compute the area A between the two melodies Q' and R
           A ←Σ_{α=1}^{k} w_α | PI_Q'[i_α]−PI_R[j_α] |
           If A < MinArea then MinArea ← A
           // Shift the query sequence
           shift-step ← min_{1≤i≤m} step[i]
           For i ← 1 to m do
               start_Q'[i] ← start_Q'[i] + shift-step
               If step[i] = shift-step then
                     after[i] ← after[i] + 1
                     step[i] ← start_R[after[i]] − start_Q'[i]
               else step[i] ← step[i] − shift-step
```

Partial shifting steps of coarse alignments for the example shown in Fig. 5 are listed in Table 1. The top 5% minimal areas yielding by these coarse alignments occur at j = 14 and 39; while the global minimal area occurs around j = 14. In the while-loop of step 5 of the above algorithm, an approximate minimal area is searched over the interval [$\delta(j–1)$, $\delta(j+1)$]. For j = 14, the searched interval is [$\delta(13)$, $\delta(15)$]= [1680, 2040].

Table 1. Shifting step δ and corresponding area $A(\delta)$ for each step of coarse alignment

j	1	2	3	4	5	…	12	13	**14**	15	16	…	37	38	**39**
δ	0	120	240	480	600	…	1560	1680	**1920**	2040	2160	…	5400	5520	**5760**
$A(\delta)$	1700	3430	3500	2940	2990	…	3830	4400	**530**	3120	4050	…	4320	4660	**1450**

3 Experimental Results

Due to the fact that people perceive the outer voice better than the inner voice in polyphonic music [16], we extract the highest notes of polyphonic music as features. Our music retrieval system first employs the pitch contour extraction algorithm to transfer the multi-tracks polyphonic music to multi-track monophonic music for both query and reference.

We used a corpus of 807 classical music songs obtained from the internet, totally consisting of 3,065,420 music notes. 50 songs were randomly chosen, the most well-known theme [17] from each is extracted to form a query containing 10 to 36 music notes. As a result, a query set of 50 queries is formed. In order to simulate some possible errors introduced by the users, we generated a variant version of the original query set by randomly changing the pitches of some notes within an octave, deleting some notes, or inserting some notes in each query. The number of alterations in each query is between 2 and 8.

To evaluate the performance of the proposed methods, we implemented and compared the naïve geometric matching algorithm proposed by D. Ó. Maidín with and without the branch-and-prune mechanism, called ***Pitch-GeoMatching*** and ***B&P-Pitch-GeoMatching***, respectively, and the proposed geometric matching algorithms with and without the branch-and-prune mechanism, called ***Pitch-Interval-GeoMatching*** and ***B&P-Pitch-Interval-GeoMatching***, respectively. The test results from the four versions of matching algorithms over the query set and its variant are shown and compared in Fig. 6 and Fig. 7, respectively. As discussed in the previous section, the time complexity of the four algorithms are all $O(m^2n)$, but with different leading constants. Table 2 shows the average search time required by each of the four algorithms tested the two query sets. The ratio of the average time costs required by the four matching algorithms is about 100: 38: 33: 18. Our ***Pitch-Interval-GeoMatching*** indeed improves the time efficiency of D. Ó. Maidín's ***Pitch-GeoMatching*** by about 3 times.

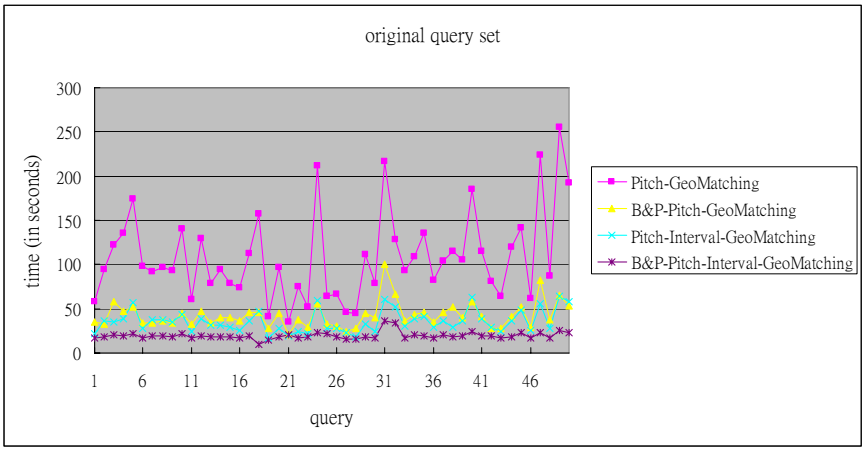


Fig. 6. Time cost required by the four algorithms tested on the original query set

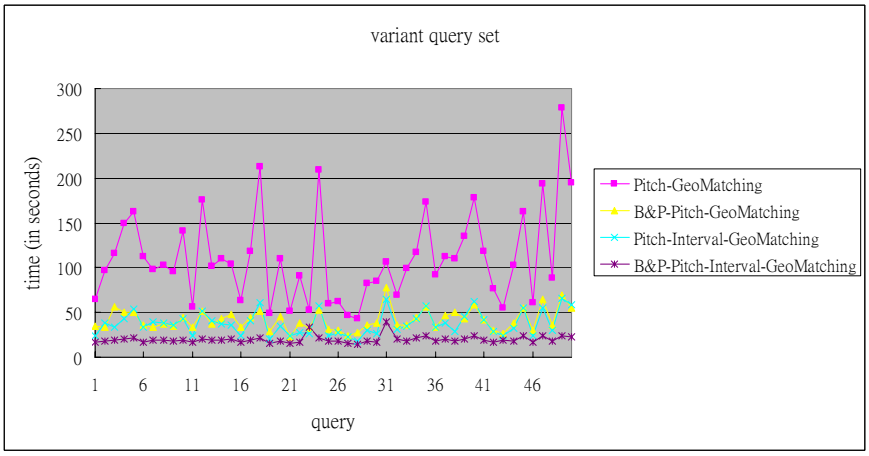


Fig. 7. Time cost required by the four algorithms tested on the variant query set

Table 2. Comparison of average time costs (in seconds) required by the four algorithms on two query sets

Algorithms / Query set	***Pitch-GeoMatching***	***B&P-Pitch-GeoMatching***	***Pitch-Interval-GeoMatching***	***B&P-Pitch-Interval-GeoMatching***
original	108.54	42.57	35.62	19.54
variant	110.91	41.76	37.76	19.82
average	109.73	42.16	36.69	19.68

More experiments were conducted to test effectiveness of the four algorithms. The retrieval rank for the query is used to evaluate the effectiveness of a matching algorithm, which is the rank of the relevant references in the retrieved list produced by the algorithm. Tested on the original query set, the retrieval ranks by each of the four algorithms are all 1. This indicates that our approach have the same effectiveness as the naïve geometric matching algorithm proposed by D. Ó. Maidín on the original query set.

Tested on the variant query set, the average retrieval ranks from the pitch matching algorithms with/without branch-and-prune are both 1.06; while the average retrieval ranks for the proposed algorithms with/without branch-and-prune are 1.48 and 1.52. The retrieval ranks of our method is slightly lower because alteration of a note in a query affects two pitch intervals in its pitch interval sequence, but only affects one pitch in its pitch sequence. However, the sacrifice of a little accuracy to improve time efficiency is worth. The retrieval rates on top-*n* lists for the four algorithms are illustrated in Fig. 8. The retrieval rates on top-3 retrieval for the ***Pitch-GeoMatching*** and ***B&P-Pitch-GeoMatching*** both achieve 100%; while the retrieval rates for the ***Pitch-Interval-GeoMatching*** and ***B&P-Pitch-Interval-GeoMatching*** achieve 100% on top-7 and top-8 lists, respectively.

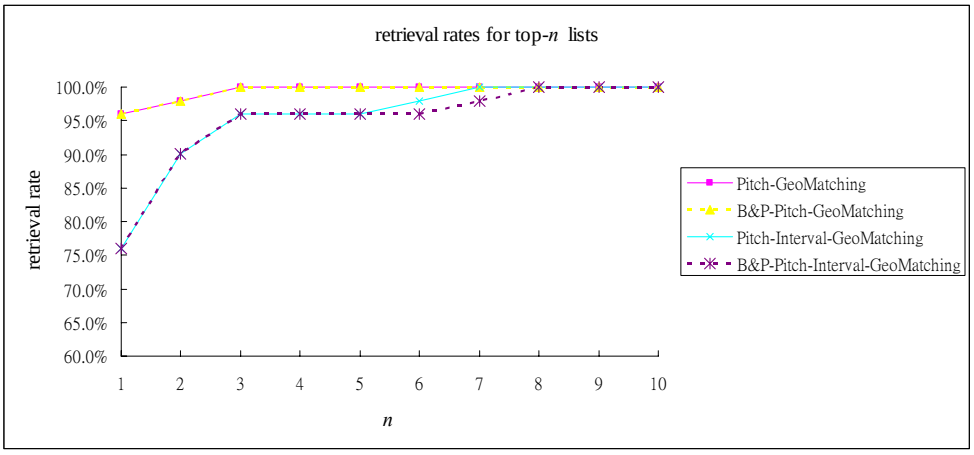


Fig. 8. Retrieval rates on top-*n* lists for pitch/pitch-interval geometric matching algorithms with/without branch-and prune mechanism tested on the variant query set

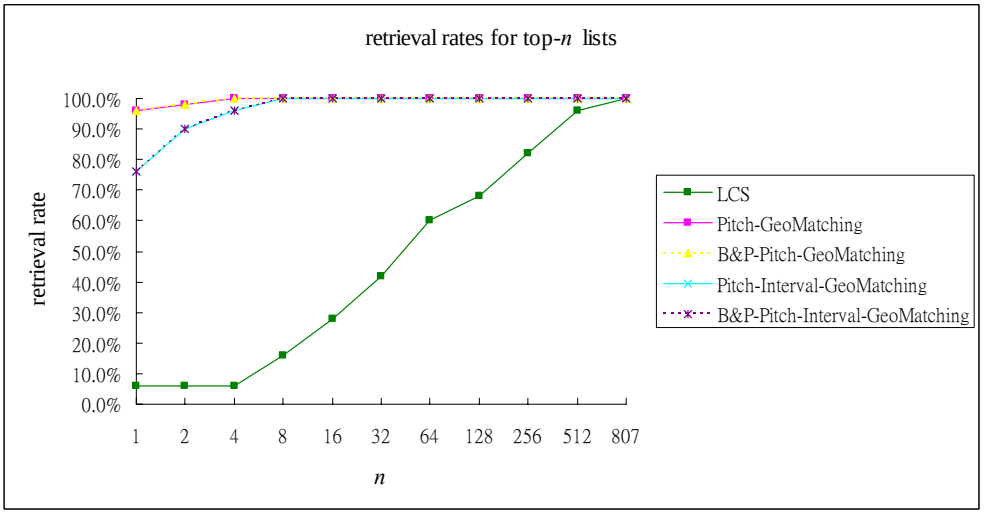


Fig. 9. Retrieval rates on top-*n* lists for LCS and our proposed methods

In order to demonstrate the effectiveness of geometric matching, we compare our method with the commonly used non-geometric matching, the Longest Common Substring (LCS) [18] method. As shown in Fig. 9, the retrieval rates of the four algorithms all achieve 100%; while that of the LCS method achieves 16% only. Notice that the LCS method is only about 2 times faster than the ***B&P-Pitch-Interval-GeoMatching***, although the former and the latter have different orders of time complexity, $O(mn)$ and $O(m^2n)$, respectively.

4 Conclusions and Future Work

Although geometric matching methods are more effective than others, they are quite time-consuming. We presented an improved version of the geometric matching method proposed by D. Ó. Maidín [13] so that its time efficiency is satisfactory for on-line applications. The improvement is achieved in two aspects: (1). matching the pitch interval instead of the pitch sequence to achieve key invariance, and (2). providing a branch-and-prune mechanism to quickly discard most of the unlikely positions of the query. Our experimental results show that the time efficiency was indeed improved with a slight drop in accuracy.

However, the proposed matching algorithms do not deal with the scaling problem of duration or rhythm, which will be the subject of our future research. Furthermore, the issue of polyphonic music matching is another problem in the field of music information retrieval, which will also be included in our future research.

5 Acknowledgement

This work was supported in part by the National Science Council of Taiwan, R.O.C. under the grant NSC-96-2221-E-032-049.

References

[1] C. Francu and C. G. Nevill-Manning, "Distance metrics and indexing strategies for a digital library of popular music," *Proceedings of the IEEE International Conference on Multimedia and EXPO*, pp. 889-892, 2000.

[2] D. Byrd and T. Crawford, "Problems of music information retrieval in the real world," *Information Processing and Management*, Vol. 38, pp. 249-272, 2002.

[3] L. A. Smith, R. J. McNab, and I. H. Witten, "Sequence-based melodic comparison: A dynamic programming approach," *Computing in Musicology*, Vol. 11, pp. 101-117, 1998.

[4] J. Foote, "An overview of audio information retrieval," *Multimedia Systems*, Vol. 7, No. 1, pp. 2-10, 1999.

[5] M. Mongeau and D. Sankoff, "Comparison of musical sequences," *Computers and the Humanities*, Vol. 24, pp. 161-175, 1990.

[6] D. Bainbridge, M. Dewsnip, and I. H. Witten, "Searching digital music libraries," *Information Processing and Management* , Vol. 41, No. 1, pp. 41-56, 2005.

[7] S. Doraisamy and S. Rüger, "A Polyphonic Music Retrieval System Using N-grams," *Proceedings of the Fourth International Conference on Music Information Retrieval*, 2004.

[8] S. Downe and M. Nelson, "Evaluation of a simple and effective music information retrieval method," *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 73-80, 2000.

[9] J.-S. Roger Jang and H.-R. Lee, "Hierarchical filtering method for content-based music retrieval via acoustic input," *Proceedings of the 9th ACM Multimedia Conference*, pp. 401-410, 2002.

[10] F. Wiering, R. Typke, and R. C. Veltkamp, "Transportation distances and their application in music-notation retrieval," *Computing in Musicology*, Vol. 13, pp. 113-128, 2004.

[11] R. Clifford, M. Christodoulakis, T. Crawford, D. Meredith, and G. Wiggins, "A fast, randomised, maximal subset matching algorithm for document-level music retrieval," *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, pp. 150-155, 2006.

[12] Y. Zhu, M. Kankanhalli, and Q. Tian, "Similarity matching of continuous melody contours for humming querying of melody databases," *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, pp. 249-252, 2002.

[13] D. Ó. Maidín, "A geometrical algorithm for melodic difference," *Computing in Musicology*, Vol. 11, pp. 65-72, 1998.

[14] G. Aloupis, T. Fevens, S. Langerman, T. Matsui, A. Mesa, Y. Nunez, D. Rappaport, and G. Toussaint, "Algorithms for computing geometric measures of melodic similarity," *Computer Music Journal*, Vol. 30, No. 3, pp. 67-76, 2006.

[15] A. Lubiw and L. Tanur, "Pattern matching in polyphonic music as a weighted geometric translation problem," in *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, pp. 289-296, 2004.

[16] J. Kerman, *Listen*, Worth Publishers, 1976.

[17] H. Barlow and S. Morgenstern, *A dictionary of musical themes*, Crown Publishers, Inc., New York, 1975.

[18] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms, 2nd Ed.*, the MIT Press and MacGraw-Hill Book Company, 2001.