A Top-down Approach to Melody Match in Pitch Contour for Query by Humming *

Xiao Wu, Ming Li, Jian Liu, Jun Yang, and Yonghong Yan

Institute of Acoustics, Chinese Academy of Sciences.
{xwu, mli, jliu, yyan}@hccl.ioa.ac.cn

Abstract. In this paper, a novel frame-based algorithm called recursive alignment(RA) for query-by-humming(QBH) application is presented. Compared with other approaches, RA optimizes melody alignment problems in a top-down fashion which is more capable of capturing long-distance information in human singing. Three RA variations which run much faster at the expense of less accuracy are presented, and they can be used as filters of QBH systems. A QBH system built upon RA and its variations is also described briefly. The experiment results show that the proposed algorithm compares favorably with other methods we have implemented.

1 Introduction

Nowadays, more and more people obtain music via internet. Traditional music search engines are usually based on metadata, and the user must type in text information to get the intended song. In some situations, however, this text interface is neither convenient nor friendly. Imagining when you have tried hundreds of query words in the search bar but the target song never jumps out, just because you forget the name of the song. This awkward situation can be solved with the help of the QBH system, which allows the user to obtain the intended song via a few seconds' humming.

A lot of research papers on monophonic QBH have been published in the past decade and many demo systems have been developed [1]. These approaches can be roughly categorized into note-based and frame-based. Most research have focused on note-based approaches which firstly segment the pitch sequence into notes and then use certain technique to find out the most similar section in the database. With a distance metric defined, approximate string matching [2] are usually chosen as the solution for computing the score between query and candidates [3–5]. Another way to align sequences is to use the Viterbi decoding in Hidden Markov Model(HMM). Meek and Brimingham [6], whose system was able to compensate many types of segmentation errors and singing errors, novelly introduced HMM framework to model errors for QBH. On the other hand,

* This work is (partly)supported by Chinese 973 program (2004CB318106), National Natural Science Foundation of China (10574140, 60535030), and Beijing Municipal Science & Technology Commission(Z0005189040391)

some investigators believe that fragile note-segmentation greatly limits the performance of note-based approaches and the grace notes in the database may also deteriorate the note matching, so they concentrate on aligning the query and target directly at the frame level. J.Jang who is a pioneer of this direction used two-stage dynamic time wrapping(DTW) and achieved 78% top-1 ratio in a 800 song database [7]. This result outperformed all the the other QBH systems at that time on similar tasks. Mazzoni and Dannenberg [8] formally addressed the deficiency of note-based approaches. The performance comparison of the three most popular melody matching algorithms, the note-level string matching, the note-level Viterbi decoding with HMM and the frame-level DTW, can be found in [9].

Notice that in all the existing research mentioned above, match score computing between query and target works in a bottom-up fashion. For the reason of having the mathematical guarantee to reach global optimal [10], almost all the QBH systems, both note-level and frame-level ones, select DP-based approach such as longest common substring(LCS), Viterbi decoding and DTW as the final stage match algorithm. DP is a typical bottom-up optimizing algorithm, which according to Cormen [11], solves problems by combining the solutions of subproblems. In music retrieval, however, the similarity between query and target not only depends on the accumulation of local match scores, but also depends on the contour in the global view. Long-distance information such as duration and rhythm is difficult to capture with bottom-up methods. J.Jang in [12] showed linear scaling(LS) worked significantly better than Viterbi decoding in their HMM-based QBH system. Their work is the first, as far as we know, to abandon bottom-up tradition in the final scoring stage. However, the LS algorithm is somewhat crude and works well only if the model is trained with considerable data.

In this paper, we propose a novel frame-based melody matching algorithm called recursive alignment (RA). RA is inspired by LS but alleviates its limitation. RA is a top-down algorithm because it first tries to match the shape of query and target in a global view and then makes finer local tuning, which works in a different way compared with DP. Just as its name implies, RA optimizes the alignment problem recursively. RA also computes the match score directly in pitch contour and it needs no training data at all. In addition, We present three variations of RA which run much faster at the expense of losing some accuracy and they can be used as filters whose responsibility is to block most of the unlikely candidates efficiently. Furthermore, we describe the whole system built upon RA and its variations. The system employs 8 level filters to select 600 most probable melody sections out of millions. Although some filters utilize some information from note-segmentation, the system does not rely much on it. While designing the filters, we are inclined to choose global features instead of those local ones with the intension of following top-down principle. In the end, we will give experiment results which show our system can achieve 83.77% top1 retrieval rate while each query can be processed in about 3.3 seconds.

Section 2 describes the existing art and RA algorithm. Section 3 presents three RA variations. Section 4 describes the QBH system built upon RA and gives experiment results of the system.

2 Melody Match at Frame Level

This section mainly describes the matching algorithm used in frame-based music retrieval systems. Sect.2.1 defines the problem. Sect.2.2 briefly introduces the existing frame-based matching algorithms such as DTW and LS. Sect.2.3 presents our proposed approach in detail.

2.1 Problem Definition

In a frame-level QBH system, the humming stream is firstly put into a pitch tracker frame by frame and then the output pitch sequence is converted to semitone scale. In this paper the pitch sequence at semitone scale is marked as $(q_1, q_2, \cdots, q_n)$. After that, guided by certain rule, searching could be started. For the consideration of runtime efficiency, indexing and filtering are usually applied to reduce the candidates. When the steps above are done, we get a set of candidates $(C_1, C_2, \cdots C_N)$ each of which is composed of a number of notes $((p_1, d_1), (p_2, d_2), \cdots, (p_m, d_m))$ where p_i and d_i refer to the pitch and duration of the ith note respectively[1]. Now the problem of melody matching could be defined: given query frame sequence $Q = (q_1, q_2, \cdots, q_n)$, note sequence $N = ((p_1, d_1), (p_2, d_2), \cdots, (p_m, d_m))$ and distance function $dist(q, (p, d))$, how to compute the match score S between the two sequences. In most cases, optimal or suboptimal alignment score is used as the match score. Once such an algorithm of melody matching is available, the top-N candidates can be found by sorting the scores of all the possible candidates.

2.2 Existing Frame-based Matching Algorithms

Dynamic programming [10] is very popular in the filed of music retrieval and is selected as the final scoring algorithm by most QBH systems. DTW, which belongs to DP category, is a frame-to-frame algorithm to find the optimal alignment path. In DTW, candidate N is expanded to a frame sequence $N^{'} = (p^{'}_1, p^{'}_2, \cdots, p^{'}_l)$ according to the duration of each note. $D_{i,j}$ which represents the minimum cumulative cost up to the ith frame in the query and the jth frame in the candidate can be computed in $O(n \cdot l)$ time with the following formula:

$$D_{i,j} = d(q_i, p^{'}_j) + min(D_{i-1,j-1},\ D_{i-1,j-2},\ D_{i-2,j-1}) \quad (1)$$

However, DTW may suffer from losing the global structure while optimization. For example, in Fig.1 where each line represents an alignment path, l_1 and l_2

[1] For simplicity, we do not distinguish the representation of p and d for different candidates.

are intuitively more reasonable than l_3 and l_4, since people rarely change their singing tempo frequently. This kind of information is usually difficult, or at least computational expensive, to capture with DP. Although many DTW variations have been developed to get better control over the long-distance structure [7, 8, 13], the natural bottom-up framework limits the performance of DTW.

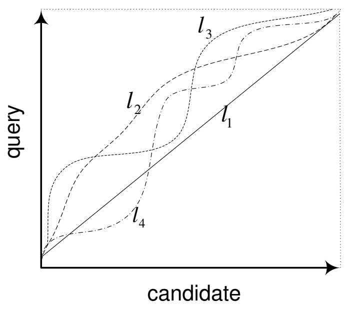

Fig. 1. Examples of alignment between query and candidate

Linear Scaling, which is depicted in Algorithm 1, is another way to match melody and needs only $O(n)$ time. Unlike DTW which builds the global solution upon local ones, LS simply chooses l_1(see Fig.1) as the alignment path. Though LS seems to be a little crude, J.Jang's research [12] showed it worked much better than Viterbi decoding with well trained HMM. The deficiency of LS is also quite obvious: local mismatch in rhythm may deteriorate the global matching, and it needs training to capture human's singing habits.

Algorithm 1 $LinearScaling(Q, N)$

```
1: INPUT: $Q = (q_1, q_2, \cdots, q_n)$, $N = ((p_1, d_1), (p_2, d_2), \cdots, (p_m, d_m))$
2: $i \leftarrow 1, j \leftarrow 1$
3: $Dura \leftarrow \sum_{k=1}^{m} d_k,\ Dist \leftarrow 0,\ Scale \leftarrow n/Dura$
4: for all $i$ such that $1 \le i \le m$ do
5:     $t \leftarrow j + Sacle \cdot d_i$
6:     $Dist \leftarrow Dist + \sum_{k=j}^{t} dist(q_k, p_i)$
7:     $j \leftarrow t + 1$
8: end for
9: return $-Dist$
```

2.3 The New Top-down Matching Algorithm

Our new algorithm called recursive alignment(RA) which is described in Algorithm 2 roots from LS but alleviates its restrictions. This algorithm differs from

DP because it starts optimization from a global view, and it also differs from LS because it tries to tune local matching recursively in order to optimize the alignment.

Algorithm 2 $RecursiveAlign(Q, N, S, D)$

```
1: INPUT: $Q = (q_1, q_2, \cdots, q_n)$, $N = ((p_1, d_1), (p_2, d_2), \cdots, (p_m, d_m))$
2: INPUT: $S = ((sx_1, sy_1), (sx_2, sy_2), \cdots, (sx_n, sy_n))$
3: INPUT: $D$ {maximum recursion depth allowed, usually set to 3 or 4}
4: $i \leftarrow 0, j \leftarrow \lfloor m/2 \rfloor, maxScore \leftarrow -\infty$
5: $sc \leftarrow \sum_{l=1}^{j} d_l / \sum_{l=1}^{m} d_l$
6: $N_1 \leftarrow ((p_1, d_1), \cdots, (p_j, d_j)), N_2 \leftarrow ((p_j, d_j), \cdots, (p_m, d_m))$
7: for all $(sx, sy)$ in pair set $S$ do
8:     $k \leftarrow \lfloor sx \cdot sc \cdot n \rfloor$
9:     $Q_1 \leftarrow (p_1, \cdots, p_k), Q_2 \leftarrow (p_k, \cdots, p_n)$
10:    $score \leftarrow LinearScaling(Q_1, N_1) + LinearScaling(Q_2, N_2)$
11:    if $score$ is larger than $maxScore$ then
12:        $maxScore \leftarrow score$
13:        $i \leftarrow k$
14:    end if
15: end for
16: if $D$ equals to 0 then
17:    return $maxScore$
18: else
19:    $Q_1 \leftarrow (p_1, \cdots, p_i)$, $Q_2 \leftarrow (p_i, \cdots, p_n)$
20:    return $RecursiveAlign(Q_1, N_1, S, D-1) + RecursiveAlign(Q_2, N_2, S, D-1)$
21: end if
```

The basic idea of RA comes from the fact that the query and the target are similar if and only if they roughly share the same shape in the global view. Usually the optimal alignment is a nonlinear transformation from query to target. This can be approximated by dividing the target section into two parts and each half uses its own linear scale[2]. The near-optimal scale factors for the two halves can be obtained through a brute-force search, that is, given a finite set of possible scale pairs $\{(sx_1, sy_1), (sx_2, sy_2), \cdots, (sx_n, sy_n)\}$, each pair is evaluated and the one with minimum distance is selected as the result. Once the top-level scale pair is decided, the alignment problem is divided into two subproblems and they can be solved with the same steps described above. Notice that during the RA optimization, we make the higher level decision ahead of lower ones. In this way, the alignment problem can be optimized in a top-down fashion recursively. Fig.2 gives an example of the optimization process.

As shown in Algorithm 2, RA utilizes LS as a subroutine. An important point should be noticed is the distance function $dist(q, p)$ we used in LS is

$$dist(q, p) = min\left\{(q-p)^2\,,\ \ floor\right\} \tag{2}$$

[2] In fact, the scale of one half is decided by that of the other half.

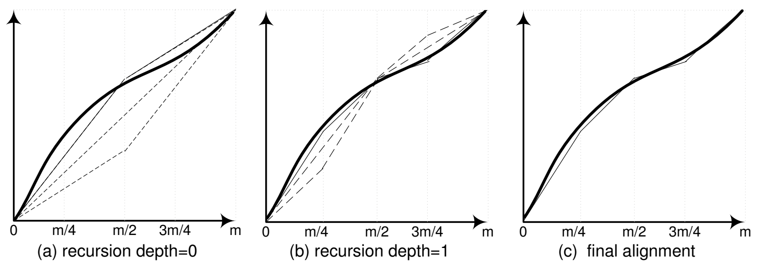

Fig. 2. An example of RA($|S| = 3$, $D = 1$). The bold line, the solid line and the dash line represent the real path, the best path in current level and the discarded path respectively.

where *floor* is a predefined constant to limit the contribution of any single aligned pair. Our study shows that the introduction of *floor* greatly increases the performance of the algorithm. A possible explanation could be: using the Euclidean distance with the form $y = \sum x^2$, unmatched pairs play far more significant roles than matched ones and introducing *floor* can reduce the impact caused by singular pairs.

Now we discuss RA in terms of graph search. Although RA dynamically extends its search space according to the previous decisions, the entire search graph can be determined if Q,N,S and D are known. As is shown in Fig.3, each path in the graph is a folded line of 2^{N+1} sects. With proper S, paths like l_3 or l_4 in Fig.1 will not be allowed, for the higher level scales which are determined ahead of the local ones restrict the local tuning within a reasonable range. In generated search graph, RA can be viewed as a heuristic search with a global pruning strategy. Here an attractive step is to use a classic left-to-right search algorithm such as DP to search in the graph generated by RA, which seems to combine the advantages of promising to find the optimal path and confining the search with human singing habits. However, our experiment results in the latter section show that pure RA performs slightly better than the hybrid algorithm which may imply that RA's top-down pruning strategy is quite effective.

The computational complexity of the proposed algorithm can be estimated quite straitforwad. At each recursion, RA performs LS for each possible scale in S, so the time complexity is $O(k \cdot n)$ where k is a constant equal to $D \cdot |S|$. In practice, RA could be faster because only few frames need to be recomputed when the scale factor changes.

3 Variations of the RA

This section will present three variations of RA. The motivation of this section is to find out faster algorithms based on the original RA at the expense of some accuracy loss. In our QBH system to be described in Sect.4.1, these variations are used as filters which are responsible for blocking most of the unlikely candidates

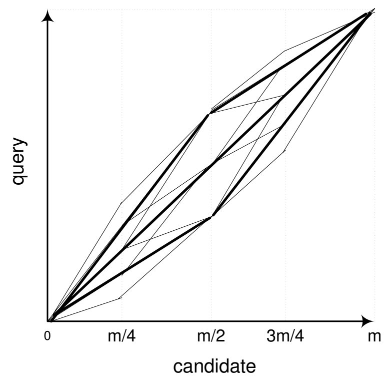

Fig. 3. Search space generated by RA ($|S| = 3$, $D = 1$)

efficiently. As the bottleneck of RA is the frame by frame computation in LS, the first two variations attempt to make LS work in note-level while the third variation attempts to compute the score of several frames in parallel.

3.1 Variation I

This variation attempts to compute the LS at note-level. Notice the frame-level summation $\sum_{k=j}^{t} dist(q_k, p_i)$ in $line6$ of Algorithm 1 which calculates the scores of all query frames aligned to the ith note. This summation could be rewritten as

$$\sum_{k=j}^{t} (q_k - p_i)^2 = (\sum_{k=1}^{t} q_k^2 - \sum_{k=1}^{j-1} q_k^2) - 2p_i(\sum_{k=1}^{t} q_k - \sum_{k=1}^{j-1} q_k) + (t-j) \cdot p_i^2 \qquad (3)$$

on the condition that the $floor$ value in (2) is ignored. The value of (3) can be obtained immediately if we have pre-computed $\sum_{i=1}^{l} q_i$ and $\sum_{i=}^{l} q_i^2$ for all l. This idea is inspired by Viola who introduced a similar skill to avoid repeat computation in human face detection [14]. By applying this, the time complexity can be reduced from $O(kn)$ to $O(km)$ where $m \ll n$. The drawback is that the singular points may badly deteriorate the performance since the distance floor is ignored.

3.2 Variation II

The second variation also computes LS at note level just like the first one does except that it utilizes information from note-segmentation. Although note-segmentation is error prone, this variation is quite insensitive to these errors. Assume query Q is segmented into r notes $((p_1', d_1'), (p_2', d_2'), \cdots, (p_r', d_r'))$, then a note-to-note RA algorithm can be used to compute the match score. As the score computation between note pair $(p^{'}, d^{'})$ and (p, d) are performed no more than $m + r$ times, the time complexity is limited to $O(k(m + r))$. Moreover, the distance floor in (2) can also be considered. The drawback is that some information is lost when the real-value frames are quantified into notes.

3.3 Variation Ⅲ

The third variation works differently from the previous two. It first divides the frequency into b sub-bands and quantifies every frame to 0 or 1 in each band which is shown in Fig.4. Then RA is performed in each band and the final score is the score summation of all bands. The intention of this variation is to avoid expensive floating-point computation, and further more, to make score calculation parallel with bit-operation utilizing the 32-bit bandwidth of modern computers. The complexity of this variation is $O(kbn/32)$ and it runs extremely efficiently if the codes are carefully programmed. The drawback is that a lot of information is lost during the binarization.

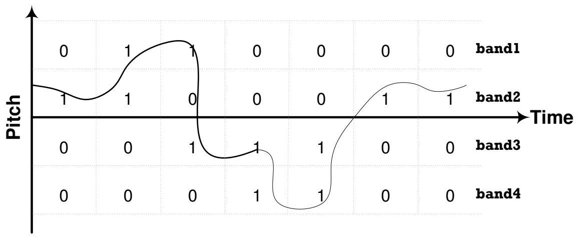

Fig. 4. RA variation Ⅲ

4 Performance Evaluation

4.1 System description

The framework of our QBH system is presented in Fig.5. The frame size is 25 ms frames with 15 ms overlapping. The pitch tracker we used here is based on an improved version of sum harmonic algorithm [15]. Note segmentation is based on sum harmonic energy and detailed description can be found in [16]. The first a few segmented notes are passed to the previous five filters to filter 8000 melody sections out of millions. Then the last three filters use all the query notes to generate the 600 most probable candidates. At last, RA is performed on all the survivals for rescoring.

Table 1 lists the eight filters and the final rescorer used in the system. The idea of the cascade filters comes from Viola [14] who used about 20 level filters to detect human faces. In fact, each filter in Table 1 can be viewed as a classifier

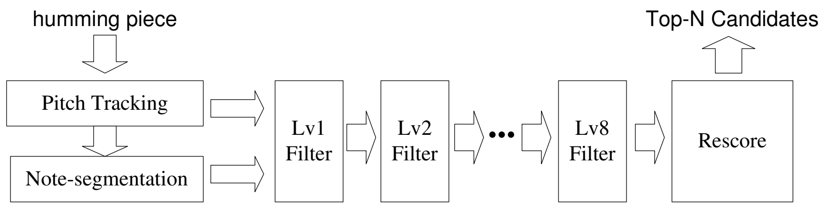

Fig. 5. System architecture

with very high detect rate, and generally, the former filters are more efficient but less accurate than the latter ones. To follow the top-down principle, we also prefer global features such as global highest pitch and histogram to local ones while designing the filters.

There are two points need to be mentioned. Firstly, to compensate errors from note-segmentation, we add very loose limit on the range of target notes: if the query has r notes, then all the sections with notes ranging from $0.5r$ to $1.8r$ are considered to be acceptable. Meanwhile, the RA VarII who utilizes note-level information of query is also insensitive to segmentation errors. So the note-segmentation may not affect the system performance too much. Secondly, since the query and the candidate are usually from the different keys, we always subtract their own mean pitch during the score computation. This simple scheme works well in most cases indeed.

Table 1. Cascade classifiers

Level	Feature(s)	Global	Source	Pass	Description
1	Real valued frames		First 3-7 notes	25%	RA VarII and Index
2	Binary quantified frames	$\surd$	1^{st}-14^{th} notes	30%	RA VarIII
3	Variance, highest pitch, etc.	$\surd$	1^{st}-14^{th} notes	35%	Linear classifier
4	Real valued frames	$\surd$	1^{st}-14^{th} notes	25%	RA VarI
5	Segmented notes	$\surd$	1^{st}-14^{th} notes	8000	RA VarII
6	Variance, highest pitch, etc.	$\surd$	All notes	50%	Linear classifier
7	Pitch histogram	$\surd$	All notes	40%	Linear classifier
8	Segmented notes	$\surd$	All notes	600	RA VarII
$*$	Real valued frames	$\surd$	All notes	TopN	Rescore with RA

4.2 The Song Database and the Test Data

The test data are 875 wave files recorded with 8000 Hz sample rate and encoded in 16bit. They are hummed by 15 average singers. Each singer is required to select at least 15 out of 36 song pieces and hum each one for three times. Roughly the 875 waves have a flat distribution over the 36 pieces.

The database contains 1180 songs with midi format and each song has 400 notes in average. Most of the songs in the database are Chinese pop songs. As we find in practice that most people shorten the long notes when they sing, we compress the duration d which is longer than the song average duration $\overline{d}$ to

$$\overline{d} + log\left[1 + \alpha(d - \overline{d})\right]/\alpha$$

where α is a predefined constant.

4.3 Experiment Results

It is really difficult to compare different QBH systems, as the retrieval performance is influenced by many factors such as test set, recording condition, database composition and so on. Still, we implement some existing approaches and attempt to make comparison in identical test condition.

Table 2. Evaluation results

	RA[*]	LS	DTW(62ms)	DTW(10ms)	RAgraph+DP
Top1	83.77%	68.40%	80.46%	82.17%	83.43%
Top2	86.17%	72.57%	83.09%	83.31%	85.94%
Top3	87.66%	75.09%	84.23%	84.34%	87.77%
Top4	88.46%	76.91%	84.91%	85.26%	88.23%
Top5	89.26%	77.94%	85.60%	86.29%	88.91%
Top6	89.60%	78.86%	86.40%	86.97%	89.26%
Top7	89.60%	79.77%	87.09%	87.77%	89.71%
Top8	90.17%	80.23%	87.43%	88.23%	89.83%
Top9	90.17%	81.03%	88.00%	88.69%	89.94%
Top10	90.29%	81.71%	88.23%	88.91%	90.17%
RunTime	3.3sec	2.9sec	3.3sec	10.5sec	off-line

[*] With four recursions and ten possible scale pairs. ($D = 4, |S| = 10$)

The QBH system is evaluated with the test data described above in the 1180 song database. We have implemented frame-based DTW [7] and LS [12] for comparison. Aimed to find the utmost of DTW, we also implemented a more accurate but less efficient DTW which uses 10 ms frame other than the 62.5 ms frame in J.Jiang's system. For the reason of impartiality, RA is replaced with DTW or LS only in the rescoring stage while leaving all the filters unchanged. Experiment results in Table 2 suggest that the RA algorithm works better than either DTW or LS. To evaluate the validity of RA pruning, we implemented a hybrid algorithm which utilizes RA to generate search space and utilizes DP to find the optimal path(see Sect.2.3). The last column of Table 2 presents the results of the hybrid algorithms. As shown in Table 2, RA performs slightly better than the hybrid algorithm, which implies the RA global pruning strategy is quite effective.

Table 3 present results of RA tested on queries with different duration. The results show that the system performs well if query is between 7 and 15 seconds. What puzzles us is the performance declines when the queries are longer than 13 seconds, which seems to be counter intuition. The explanation could be one of the following two: (1) The filters are not robust enough with long query. (2) The RA algorithm gets less powerful when query duration increases. By analyzing the errors, we found most of the missed queries are blocked by filters, which implis the former explanation may be more appropriate.

Table 3. Results of RA Tested on queries with different duration

Duration	Count	Percent	Top1	Top5	Top10
$<$ 5 sec	30	3.43%	63.33%	76.67%	76.67%
5 - 7 sec	272	31.09%	77.21%	88.97%	90.07%
7 - 9 sec	277	31.66%	86.64%	90.61%	91.70%
9 - 11 sec	142	16.23%	83.80%	90.14%	90.14%
11 - 13 sec	43	4.91%	90.70%	93.02%	93.02%
13 - 15 sec	57	6.51%	89.47%	91.23%	91.23%
$\geq$ 15 sec	54	6.17%	83.33%	83.33%	88.89%

5 Conclusions

In this paper, we proposed a novel frame-based algorithm called recursive alignment(RA) for music retrieval. Compared with existing art, RA optimizes melody alignment problems in a top-down fashion which we believe are more capable of capturing long-distance information in human singing. We also presented three variations of RA which run much faster at the expense of less accuracy and they can be used as filters of QBH systems. Moreover, we briefly described our whole system built upon RA and its variations. We believe that melody matching should start from global view, which is being supported by our experiments.

References

1. Typke, R., Wiering, F., Veltkamp, R.: A survey of music information retrieval systems. In: Proc of ISMIR. (2005)
2. Lemström, K.: String Matching Techniques for Music Retrieval. PhD thesis, University of Helsinki, Finland (2000)

3. Ghias, A., Logan, J., Chamberlin, D., Smith, B.: Query by humming: musical information retrieval in an audio database. In: Proc of ACM Multimedia. (1995)
4. McNab, R., Smith, L., Witten, I., Henderson, C., Cunningham, S.: Towards the digital music library: tune retrieval from acoustic input. In: Proc of ACM international conference on Digital libraries. (1996)
5. Parker, C.: Applications of binary classification and adaptive boosting to the query-by-humming problem. In: Proc of ISMIR. (2005)
6. Meek, C., Birmingham, W.: Applications of binary classification and adaptive boosting to the query-by-humming problem. In: Proc of ISMIR. (2002)
7. Jang, J., Chen, J., Kao, M.: A query-by-singing system based on dynamic programming. In: International Workshop on Intelligent systems Resolutions. (2000)
8. Mazzoni, D., Dannenberg, R.: Melody matching directly from audio. In: Proc of ISMIR. (2001)
9. Dannenberg, R., Birmingham, W., Tzanetakis, G., Meek, C., Ning, N.: The musart testbed for query-by-humming evaluation. In: Computer Music Journal. (2004)
10. Bellman, R.: Dynamic Programming. Princeton Univ. Press (1957)
11. Cormen, T., Leiserson, C., Rivest, R.: Introduction to algorithms. MIT Press (1990)
12. Jang, J., Hsu, C., Lee, H.: Continuous hmm and its enhancement for singing/humming query retrieval. In: Proc of ISMIR. (2005)
13. Duchi, J., Phipps, B.: Query by humming: Finding songs in a polyphonic database. In: Stanford Computer Science Department. (2005)
14. Viola, P., J.M.: Robust real-time object detection. In: International Journal of Computer Vision. (2002)
15. Li, M., W.Y.Y.T.: High efficient pitch tracking method for tonal feature extraction. In: Proc of International Conference of Chinese Computing. (2001)
16. Li, M., Y.Y.: An humming based approach for music retrieval. In: Proc of National Conference on Man-Machine Speech Communication. (2005)