A PHRASE-LEVEL PIECEWISE LINEAR SCALING ALGORITHM FOR MELODY MATCH IN QUERY-BY-HUMMING SYSTEMS

Wenxiao Cao[*]*,Danning Jiang*[**]*, Jue Hou*[*]*, Yong Qin*[**]*, Thomas Fang Zheng*[*]*, Yi Liu*[*]

[*]Center for Speech and Language Technologies, Division of Technical Innovation and Development
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[**]IBM China Research Lab, Beijing, China
{caowx, houj}@cslt.riit.tsinghua.edu.cn, {jiangdn, qinyong}@cn.ibm.com, {fzheng, eeyliu}@tsinghua.edu.cn

ABSTRACT

The Query-by-Humming (QBH) system allows users to retrieve songs by singing/humming. In this paper we propose a phrase-level piecewise linear scaling algorithm for melody match. Musical phrase boundaries are predicted for the query to split it to phrases. The boundaries of melody fragment corresponding to each phrase are allowed for adjusting in a limited scope. The algorithm employs Dynamic Programming and Recursive Alignment to search for the minimal piecewise matching cost upon Linear Scaling at phrase-level. Our experimental results on 5223 melody database show that the proposed algorithm outperforms traditional algorithms. The proposed algorithm gives significant improvements of 17.0%, 14.7% and 4.8% with respect to Linear Scaling, Dynamic Time Wrapping and Recursive Alignment in top-1 rate, respectively. The results show that the proposed algorithm is more efficient than the previous algorithms.

Index Terms— *Phrase-level, Piecewise Linear Scaling, Query-by-Humming*

1. INTRODUCTION

The Query-by-Humming (QBH) system allows users to retrieve songs by singing/humming a fragment of melody. It provides a natural interface for users because in many cases the user cannot remember any key words of the song except the melody.

In recent years, some research work showed the advantage of frame-based matching approaches, in which the melody match is performed on the frame-level pitch contours instead of the note sequences [1]. The most commonly used algorithm is Dynamic Time Warping (DTW) [2], searching for the path of alignment with the minimum pitch distance. However, providing a straight-forward way to compute melodic similarities, DTW fails to measure the distance of rhythms. The path with minimum pitch distance would be invalid at all because the corresponding rhythms were very different between the query and the target melody.

To solve this problem, Jang et al. [3] proposed Linear Scaling (LS) algorithm, which simply stretches or compresses the query contour and match it point-by-point with the target contour. It assumes the query and melody are linearly correlated with each other and this is proved to be even better than HMM-based algorithm [4]. However, due to the fact that the real contour deviates from reference rhythm, the above assumption is often too strong and leads to a lot of mismatches in melody matching. Besides LS, Recursive Alignment (RA) is used to incorporate rhythm measures in similarity computing [5]. The algorithm recursively splits the query contour into two segments from the middle, performs LS match within each segment, and searches for the optimal split point in the candidate melody. Experiments reported in [5] showed that RA outperformed both DTW and LS, but it still has the drawback that no evidence shows that splitting at the middle of contour is the best way for piecewise matching.

In this paper, we propose an approach of piecewise Linear Scaling. Musical phrase boundaries are predicted and the query is split to phrases. The boundaries of melody fragment corresponding to each phrase are allowed for adjusting in limited scope. Then Dynamic Programming (DP) or RA algorithm is used to search the minimal piecewise matching cost upon LS match at phrase-level. Similar to spoken languages, music is also organized with a hierarchy structure. A "musical phrase" in this structure is a section of music that is relatively self-contained and coherent over a medium time scale [6]. While singing, the singer usually breathes at musical phrase boundaries, which often cause rhythm variations. Thus splitting the query to phrases is a better way for piecewise LS match.

The paper is organized as follows. Section 2 describes our proposed melody match algorithm. Section 3 presents the QBH system used to evaluate the algorithm. Experimental results are given in Section 4 and we draw our conclusion in Section 5.

2. THE PIECEWISE LINEAR SCALING MATCH

In a frame-based QBH system, the query signal is first processed to extract the pitch contour (in semitone scale), represented as $Q = (q_1, q_2, \cdots, q_m)$. Each candidate in melody database is represented as a sequence of notes, $S = (s_1, s_2, \ldots, s_n)$, where $s_i = (p_i, d_i), 1 \le i \le n$, p_i and d_i are the pitch and duration of s_i, respectively. After normalizing the candidate melody to the same key scale and tempo with query, the match algorithm needs to score the candidate by minimizing the cumulative pitch distance along the path of alignment between the query contour and the melody note sequence. The alignment path always starts from (q_1, s_1), but may ends at any $(q_m, s_i)(1 < i \le n)$, as user sings only a fragment of the melody. The system outputs the top-N results by sorting the match scores.

2.1. Phrase segmentation

The aim of phrase segmentation is to segment the query contour into several fragments, and each fragment roughly corresponds to a musical phrase. The singer usually breathes at phrase boundaries. Thus a relatively longer unvoiced segment is observed in the query signal. Figure 1 shows the music score and pitch contour of a fragment of the famous "Twinkle, Twinkle, Little Star" composed by Mozart. The comparison of the music score and query pitch contour shows that query phrases segmented by unvoiced segment are consistent with "musical phrases" in the score.

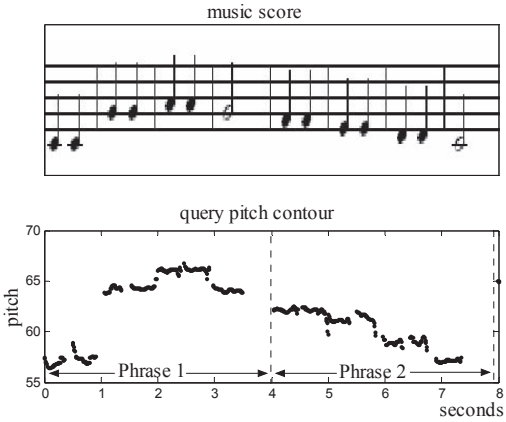

Figure 1. An example of "musical phrases"

In phrase segmentation, we first detect all unvoiced segments longer than a threshold T_{sil}, and then compose the initial set of phrases with the fragments between each two adjacent unvoiced segments. To prevent short fragments caused by frequent discontinuities, short fragments are combined with one of their neighbors by a one-pass, left-to-right combination algorithm. For each element in the initial phrase set, if its duration is longer than threshold T_{ph}, the algorithm directly moves to the next element. Otherwise, the algorithm will compare the duration of unvoiced segments before and after this phrase. If the one before it is longer, this phrase will be merged with the next phrase, otherwise it will be merged with the previous phrase.

After phrase segmentation, the query contour can be regarded as a concatenation of phrases:

$$Q = (ph_1, ph_2, \ldots, ph_k) \quad (1)$$

where each ph_i $(1 \le i \le k)$ is represented as:

$$ph_i = [q_{u_i}, q_{u_{i+1}}) = (q_{u_i}, q_{u_i+1}, \cdots, q_{u_{i+1}-1}) \quad (2)$$

where u_i is the beginning frame of ph_i, and $u_{i+1} - 1$ is the end frame.

2.2. Phrase-level piecewise linear scaling algorithm

After segmenting the query into phrases, the proposed algorithm first locates a set of candidate notes in the melody for each u_i, represented as $C(u_i)$. $C(u_1) = \{s_1\}$ for $i = 1$. For $1 < i \le k+1$, $C(u_i)$ is composed of melody notes which are close to u_i in the time axis:

$$C(u_i) = \{s_j | t_{u_i} - d_c \le \sum_{k=1}^{j-1} d_k \le t_{u_i} + d_c\} \quad (3)$$

where d_c is a duration constant to allow for a certain degree of imperfect rhythm. Figure 2 shows a query pitch contour along with the corresponding melody. The query has been segmented into two phrases: $ph_1 = [u_1, u_2)$, $ph_2 = [u_2, u_3)$, and the arrow headed line shows which notes are the match candidates of u_2 and u_3 respectively. The figure shows that $C(u_2) = \{s_4, s_5, s_6\}$ and $C(u_3) = \{s_8, s_9, s_{10}\}$.

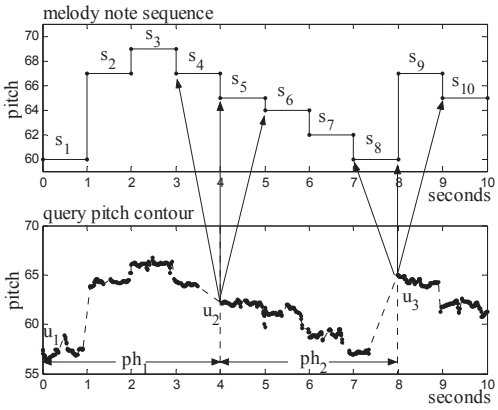

Figure 2. A demonstration of note candidates locating for query phrase boundaries

After $C(u_i)$ is set, each query phrase is assigned with several relevant melody segments as match candidates. The match cost between each phrase and its candidate is calculated with the LS algorithm, thus the alignment path within each phrase is always a straight line. The proposed algorithm searches for an optimal segmentation of melody for alignment with the query phrases and minimize the overall match cost. The overall match cost of the query and the melody is calculated as the sum of the LS match scores of each phrase and its corresponding melody fragment. Thus the optimal melody segmentation $(c_1^*, c_2^*, \cdots, c_{k+1}^*)$ should minimize the following cost function:

$$f(c_1, c_2, \cdots, c_{k+1}) = \sum_{i=1}^{k} LS([q_{u_i}, q_{u_{i+1}}), [c_i, c_{i+1})) \quad (4)$$

where $c_i \in C(u_i)$ $(1 \le i \le k+1)$ is the *i*-th note candidate for boundary u_i. $LS([q_{u_i}, q_{u_{i+1}}), [c_i, c_{i+1}))$ is the LS match score for the query phrase $[q_{u_i}, q_{u_{i+1}})$ and the melody fragment $[c_i, c_{i+1})$.

Various methods can be used for searching the $(c_1^*, c_2^*, \ldots, c_{k+1}^*)$. By employing Dynamic Programming and Recursive Alignment as search algorithms, we got two variations of the proposed algorithm: PHDP and PHRA.

2.2.1. The PHDP algorithm

The PHDP algorithm generates a cost matrix $D_{0..k,0..h}$ ($h = \max_{1 \le i \le k+1} |C(u_i)|$). The initial conditions are: $D_{0,0} = 0$, $D_{0,j} = INF$ $(1 \le j \le h)$, and $D_{i,0} = INF$ $(1 \le i \le k)$. For each $1 \le i \le k$ and $1 \le j \le |C(u_{i+1})|$, $D_{i,j}$ is calculated as:

$$D_{i,j} = \min_l (D_{i-1,l} + LS([q_{u_i}, q_{u_{i+1}}), [c_l(u_i), c_j(u_{i+1}))) \quad (5)$$

where $LS([q_{u_i}, q_{u_{i+1}}), [c_l(u_i), c_j(u_{i+1})))$ is the LS match score between the query phrase $[q_{u_i}, q_{u_{i+1}})$ and the melody fragment $[c_l(u_i), c_j(u_{i+1}))$; $c_l(u_i)$ and $c_j(u_{i+1})$ are the *l*-th and *j*-th

element in $C(u_i)$ and $C(u_{i+1})$, respectively. The final match cost is given as $\min_j D_{k,j}$.

2.2.2. The PHRA algorithm

Unlike the bottom-up fashion of PHDP, PHRA algorithm optimizes in a top-down approach. RA was first proposed in [5], and our algorithm differs from it in recursively splitting the query only at phrase boundary nearest to the middle of the query contour. Given maximum depth r, the final match score for this algorithm can be given by $\min_j RA([q_{u_1}, q_{u_{k+1}}), [s_1, c_j(u_{k+1})), 1)$, where function $RA([q_{u_i}, q_{u_j}), [s_{c_w}, s_{c_v}), d)$ is a recursive function for matching query phrase $[q_{u_i}, q_{u_j})$ with melody fragment $[s_{c_w}, s_{c_v})$ at depth d. It's calculated as follows:

(1) Calculate $S_{LS} = LS([q_{u_i}, q_{u_j}), [s_{c_w}, s_{c_v}))$;
(2) If $d = r$ or $u_j - u_i = 1$ then return S_{LS};
(3) Split the query contour into two parts at the phrase boundary u_m nearest to the middle. Then minimize the match cost:

$$S_{RA} = \min_j (RA([q_{u_i}, q_{u_m}), [s_{c_w}, c_j(u_m)), d+1) + RA([q_{u_m}, q_{u_j}), [c_j(u_m), s_{c_v}), d+1)) \quad (6)$$

(4) return $\min(S_{LS}, S_{RA})$.

The algorithm optimizes the match cost by recursively splitting the query contour until the maximal recursion depth r is reached.

3. QBH PROTOTYPE SYSTEM

We developed a QBH prototype system to evaluate the effectiveness of the proposed algorithm. The system consists of three components: pitch extraction, melody normalization, and melody match, as described in Figure 3. When a query is sent to the system, the pitch extraction is first performed to estimate the pitch contour of the query. Then, the pitch contour is used to estimate the key shift and tempo ratio between the query and each candidate in the melody database, and normalize the candidates to make them have the same key and tempo with the query. Finally, the melody match algorithm is performed to score each normalized candidate and give the top-n list.

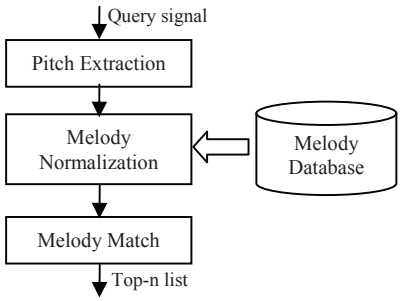

Figure 3. Paradigm of the QBH prototype system

3.1. Pitch extraction

A modified autocorrelation algorithm is used to estimate the pitch contour. Before pitch extraction, the query signal is first decoded into a sequence of notes/silence segments by performing a statistical note transcription algorithm [7]. Unlike traditional autocorrelation algorithm, the modified pitch extraction algorithm searches for the pitch related peak only in the area associated with the decoded note rather than search globally. Our previous work showed that the statistical note transcription algorithm is able to reduce the frequency errors effectively in various acoustic conditions [7]. Thus the pitch extraction algorithm also inherits this advantage and tends to be robust against noise. The pitch contour is finally converted into the semitone scale.

3.2. Melody normalization

Candidates in melody database need to be normalized to deal with different keys and tempos before melody match. In our system, this is done via a heuristic search procedure similar with that proposed in [2]. The key shift b is initialized as the pitch distance between the first decoded note of the query and the first note of the candidate. Then the following search procedure is performed to estimate the optimal tempo ratio α in $[0.5, 2.0]$:

(1) Initialization:$\alpha = 1.0$,

$$span = \begin{cases} 1.0/N, \; if \; \alpha > 1.0 \\ 0.5/N, \; if \; \alpha < 1.0 \end{cases} \quad (7)$$

where N is a preset constant for splitting the tempo ratio value interval $[0.5, 2.0]$ into $2 \times N$ candidates, witch is 10 in our experiments.
(2) Compute the distances between the query and the candidates normalized with α,$\alpha + span$, and $\alpha - span$, represented as d_0, d_1, and d_{-1}, respectively. (In our experiments, the distance was calculated with LS)
(3) If $d_0 = \min\{d_0, d_1, d_{-1}\}$ or d_0 reaches the boundary of $[0.5, 2.0]$, stop; otherwise, go to (4).
(4) Update α:
If $d_1 = \min\{d_0, d_1, d_{-1}\}$, then $\alpha = a + span$;
else if $d_{-1} = \min\{d_0, d_1, d_{-1}\}$, then $\alpha = a - span$.
Go to (2).

In our experiments, the above search procedure was accelerated by buffering the distances calculated in (2), so repeated distance calculations can be avoided. Based on the estimation of α, we continue to refine the estimation of key shift b with another heuristic search procedure. Given the estimation of key shift and tempo ratio, the candidate melody $S = (s_1, s_2, \cdots, s_n)$ is then normalized as $S' = (s'_1, s'_2, \cdots, s'_n)$, where $s'_i = (p_i + b, \alpha * d_i)$.

4. EXPERIMENTS

4.1. Database

The query dataset used in our experiments was the ThinkIT QBH query corpus, which was one of the evaluation sets of 2008 MIREX [8] QBH competitions. The query set contains 355 queries recorded by ordinary singers, most of which are sung with lyrics, and cover 106 melodies of Chinese popular songs. The average length of the queries was 12.4s, and saved as 8k Hz, 16 bit PCM files.

Our melody database consisted of two parts. The first part was 106 melodies from ThinkIT QBH melody corpus. These were the target melodies of the queries. These 106 melodies were manually segmented into 2,213 segments, each of which roughly corresponded to a musical "sentence" or "phrase". Only the

starting point of each melodic segment was regarded as a match entry. The second part of the database contained 5,117 EsAC [9] melodies as noises, and they were segmented into 29,199 segments in such a way that each line in the original EsAC melody file corresponded to one segment in the database. All together our database contained 5,223 melodies and 31,412 melodic segments.

4.2. Experimental results

In our experiments, the melody match accuracy was measured using both top-1 accurate rate and Mean Reciprocal Rate (MRR). The MRR was commonly used as a standard measurement of QBH performance in the MIREX competitions [8]. The MRR can be calculated as the average of the reciprocal rank of the target melody:

$$MRR = (\sum_{q=1}^{N} \frac{1}{c_q})/N \quad (8)$$

where c_q is the rank of the target melody for the q-th query. In the experiments, if the target melody is not in the top 20 returns, the corresponding reciprocal term is set to be 0. The value of MRR varies between 0 and 1.

Table 1 gives a comparison of melody match accuracy of the proposed algorithms (PHDP and PHRA), LS, DP, and RA. It is seen that the proposed approach outperforms the traditional methods of LS, DP, RA with 17.0%, 14.7%, 4.8% in top-1 rate, respectively. The reasons are that PHDP and PHRA base on better segmentation of query for piecewise matching. While LS is one case in DP, and RA is better in catching the global shape of the query and melody [5]. Thus DP is better than LS and RA is better than DP.

Furthermore, it is shown that PHRA outperforms PHDP. The relative increase of top-1 rate and MRR are 7% and 4%. This is because that recursive alignment optimizes in a top-down fashion, and as a result, is more capable to match the global shape of query and melody. Thus top-down fashion is better for searching phrase-level optimal piecewise matching.

Table 1. The accuracy of each melody match algorithm.

	LS	DP	RA	PHDP	PHRA
Top-1 rate	0.575	0.587	0.642	0.651	0.673
MRR	0.608	0.621	0.669	0.682	0.701

In addition, an interesting result was found in our experiment. The PHRA outperformed the RA algorithm proposed in [5], and the relative increase of top-1 rate and MRR is 4.8% and 4.8%. The reason is that the original RA algorithm always splits the query in the middle, but no evidence showed that the rhythm variations are more likely to happen in the middle than other places of the query. In contrast, the PHRA algorithm splits the query only at the estimated phrase boundaries, where the rhythm variations are more likely to happen because people often breathe in such places.

Our experiments also showed that our proposed algorithm improves the recognition accuracy without bring large additional computation. The computation time related to Table 1 for PHRA only increase 5% than RA while PHDP even reduce 6% than RA.

5. CONCLUSIONS

In this paper, we proposed a phrase-level piecewise linear scaling algorithm for melody match. In this approach, musical phrase boundaries are predicted and the query is split to phrases. The boundaries of the melody fragment related to each phrase are allowed to be adjusted in restrained scope. We applied Dynamic Programming or Recursive Alignment algorithm to search for the minimal piecewise matching cost upon LS match at the phrase-level. The experiments on melody database with 5223 melodies proved the effectiveness of proposed algorithm. It increases the top-1 rate with 17.0%, 14.7% and 4.8% compared to LS, DP, and RA, respectively.The relative increase of MRR is 15.3%, 12.9%, and 4.8% respectively. The results show that the proposed algorithm is more efficient than the previous algorithms for melody match.

6. ACKNOWLEDGMENT

We would like to thank Liu Wen of IBM China Research Laboratory, Beijing, for his help in running experiments. This work was supported by joint research grant of IBM and Tsinghua University 2007-2008.

7. REFERENCES

[1] R.B. Dannenberg, W.P. Birmingham, G. Tzanetakis, et al., "The MUSART Testbed for Query-by-Humming Evaluation," Proc. of International Symposium of Music Information Retrieval (ISMIR), 2003.

[2] J.R. Jang and M.Y. Gao, "A Query-by-Singing System based on Dynamic Programming," Proc. of International Workshop on Intelligent Systems Resolutions, pp. 85-89, Dec. 2000.

[3] J.R. Jang, H.R. Lee, M.Y. Kao, "Content-based Music Retrieval Using Linear Scaling and Branch-and-Bound Tree search," Proc. of IEEE International Conference on Multimedia and Expo, August 2001.

[4] J.R. Jang, C.L Hsu, H.R Lee, "Continuous HMM and its Enhancement for Singing/Humming Query Retrieval", Proc. of ISMIR, 2005.

[5] X. Wu, M. Li, J. Liu, et al., "A Top-down Approach to Melody Match in Pitch Contour for Query by Humming," Proc. of International Symposium on Chinese Spoken Language Processing, Dec. 2006.

[6] WikiPedia page: http://en.wikipedia.org/wiki/Phrase_(music)

[7] D.N. Jiang, M. Picheny, and Y. Qin, "Voice-melody Transcription under a Speech Recognition Framework," Proc. of ICASSP 2007.

[8] http://www.music-ir.org/mirex/2008/index.php/Main_Page

[9] http://www.cs.uu.nl/events/dech1999/dahlig/