Query By Singing/Humming System Using Segment-based Melody Matching for Music Retrieval

WEN-HSING LAI, CHI-YONG LEE
Department of Computer and Communication Engineering
National Kaohsiung First University of Science and Technology
No. 1, University Rd., Yanchao Dist., Kaohsiung City 824
TAIWAN
lwh@nkfust.edu.tw

Abstract: - Query By Singing/Humming is a melody recognition system for music retrieval by using a singing or humming query. A segment-based melody matching approach is proposed to solve the problems of puff noise and inconsistent tempo, and to lower the computational complexity of traditional Linear Scaling method in Query By Singing/Humming system. The query and midi in database are separated into several segments by four methods we proposed, namely, "Cross *n* Semitones", "Moving Average", "Composite Moving Average", and "Combination of Cross *n* Semitones and Composite Moving Average". Linear scaling is then applied in each corresponding segment. The top 10 recognition rate of our method can reach 70.6%. In addition, from our examples, it shows that segment-based method can solve the problems of puff noise and inconsistent tempo better than traditional global Linear Scaling. The segment-based methods also reduce the computation complexity by jumping to the possible segment boundary and using the segment ratio as scaling ratio, instead of comparing frame by frame from the very beginning and trying different scaling ratio in traditional Linear Scaling method.

Key-Words: - Query By Singing/Humming, Music Retrieval, Melody Matching, Segment-Based, Linear scaling, Moving Average

1 Introduction

Due to the vigorous development of the Internet, a large amount of information flow on the web, which makes information more easily accessed. However, in this flood of information, how to find the information needed becomes a major issue and challenge. Keyword search engine is created, for quickly finding the information we want. Similar situation happens on the large amount of music on the Internet. When we think of a piece of music but do not know the song title or artist, how can we find it? Query By Singing/Humming, (QBSH) [1-10] for musical retrieval is the technology developed for this demand.

QBSH is a melody recognition system by using a singing or humming query. The system will record the singing/humming query, extract the feature (e.g. pitch) from the voice, and compare the feature of singing/ humming query with the feature of the songs stored in database, and report the most similar song to the user.

Some systems combine the melody features with lyrics, [11-13], while the other focus on melody features only. For melody features, the mostly used feature for recognition is frame-level pitch, as we adopted in our system, or transcribed note [14-19], or both [20]. Pitch information is generally gotten from pitch tracking algorithm [6][21-22]. To increase the accuracy, detecting and filtering the effects of noise, vibrato, bending and inaccurate tuning by users [23] is helpful. At last, the song with the smallest feature distance [24] or the best score [25] with the query could mean that it is the most similar song and is the most possible answer to the query.

However, QBSH for music retrieval encounter problems in the reality test. The very majority users of this system are amateurs, or even without musicality at all. Their humming or singing generally has several flaws which make the music recognition or matching very difficult. For example, the pitch often drifts, or even out of tune. Sometimes the users change their singing key locally or globally. Their tempo is unstable, or they simply not follow the beat. In addition, some kind of drift is a natural phenomenon or is an expression for singing, like vibrato, that we cannot avoid as long as it is a human singing.

To conquer the problem, there are currently several techniques of QBSH, for example, hidden

Markov model (HMM) [26-27], locality sensitive hashing [28], etc. As the retrieval accuracy becomes higher, the retrieval speed becomes slower because the methods have become more sophisticated and computationally expensive. Some researches start to turn their attention to increase recognition speed by using tree-based method [29], fast Fourier transforms of note sequences [30], or GPUs [31].

Among the techniques of QBSH, the most common ones are Linear Scaling (LS), Dynamic Time Warping (DTW) [32-39] or their combinations [3][5][40]. LS stretches or compresses the query pitch contour globally and compares with the target pitch contour. However, DTW searches for the path of alignment and allows a non-linear mapping of the query to the target by minimizing their pitch distance to get the best mapping path. If we allow the paths to *[i, j]* are from *[i-2, j-1]*, *[i-1, j-1]*, or *[i-1, j-2]*, the minimum distance from *[0, 0]* to *[i, j]* is

$$\mathrm{D[i,j]} = \mathrm{d[i,j]} + \min \begin{cases} D[i-2, j-1] \\ D[i-1, j-1]. \\ D[i-1, j-2] \end{cases} \quad (1)$$

d[i,j] is the distance between sample *i* of query and sample *j* of target. DTW determine the path with the minimum *D[M,N]* as the best path. *M* and *N* are the length of query and target. Although the recognition rate of DTW is higher, the computational complexity is also higher and the recognition time is longer. LS, which simply stretches or compresses the query pitch contour globally and matches it frame-by-frame (one pitch point per frame) with the target pitch contour, is simpler. It uses more intuitive linear stretch. Its computational complexity is lower and recognition speed is faster.

But when LS encounter the situation of short time puffs, which usually appears at the beginning of humming or singing, or unstable tempo, which is also very common especially for amateurs, the recognition rate will decline because of its simple global scaling. Besides, we do not know the singing or humming starting position in songs and we do not know the exact humming tempo variation, thus we generally need to compare from the beginning frame by frame with different scaling ratio, which increases computational load. We try to solve the puff and unstable tempo problems and speed the recognition time by introducing segment-based LS approaches. Our approach is to separate query and midi in database into several segments, and then apply LS on each corresponding segment. Points with abrupt pitch change will be considered as possible segment boundaries.

There are four steps in the segment-based method we proposed. First, for both the query singing/humming and the midis in database, we select the points with abrupt change of pitch as the segment boundary candidates. Second, from the segment boundary candidates in query singing/humming, four segment boundaries, including the first, the last, the highest, and the second highest boundaries, are considered as query segment boundaries. Third, by calculating the matching distance of query segment boundaries and the segment boundary candidates in midi database and choosing the shortest distance pair, the most possible segment matching pairs are determined. Finally, we stretch each segment according to the matching pair by linear scaling. After compare all the songs in database, the song with the shortest distance is the answer to the query singing/humming.

In the first step, selecting segment boundary candidates, we propose four approaches. We call them "Cross *n* Semitones", "Moving Average", "Composite Moving Average", and "Combination of Cross *n* Semitones and Composite Moving Average". "Cross *n* Semitones" selects the points with pitch variation over *n* semitones within frames as possible segment boundary candidates. Then, considering moving out the interference of the drifts in the pitch of query singing/humming, the technique of Moving Average" is applied to smooth out short-term fluctuations, and highlight the long-term trend. Further, we found out that moving average curves with different window size overlap at area with large dynamic pitch variation. Therefore, this overlapped point can be used as a possible segment boundary candidate and we call it as "Composite Moving Average" method. The last method is a combination of Cross *n* Semitones and Composite Moving Average. If a point is selected by both "Cross *n* Semitones" and "Composite Moving Average", then it is selected as a segment boundary candidate. We will introduce the four method of how to select segment boundary candidates in details in Section 3.

We try to use the segment-based method to eliminate the recognition inaccuracy caused by puff noise and tempo inconsistency. Besides, jumping to the possible segment boundary directly and using the segment ratio as scaling ratio, instead of comparing frame by frame from beginning with different scaling ratio in traditional LS, can also reduce computational load.

This paper is organized as follows. The next section will introduce the background of LS, which we used in each corresponding segment pair. Then, the segment-based method we propose is presented in the third section. Experimental results will be shown in the following section. Finally, conclusions and future works are discussed.

2 Linear Scaling

We propose segment-based approach in this paper. Segment boundaries are predicted and the query and songs in database are split into segments for matching. Inside each matching segment pair, LS is used. To understand it more clearly, the traditional LS is introduced in this section.

LS [6][7] is also called uniform scaling or global scaling. It is a frame-based method for melody recognition. Compared with other methods, it is very straightforward. Typical LS working on frame-level pitch uses interpolation to uniformly expand or compress the pitch vector of query singing/humming, and compare the scaled pitch curve with the midis in database. The scaling ratio, which is the length ratio between the original and scaled length of pitch vector, normally is constrained to a range, for example, 0.5 to 2.0. Different scaling ratio in the range with a certain amount of step increase, e.g. 0.1, are tried and compared. That makes the scaling ratio change from 0.5, 0.6, 0.7,... to 2.0. Totally 16 different scaling ratio will be used in this case. Distances to all songs in database with different scaling ratio are calculated. The distance measure can be simply chosen as L1 or L2 norm. Generally, key transposition, which shifts the pitch to the same median in L1 norm or mean in L2 norm as that of midi song in database, are used, because key shifting is common and need to be taken into consideration. The midi in database with the minimum distance is the most likely song to the query singing/humming. Therefore, the range and the step increase of scaling ratio will have great impact on the recognition rate and computational load. Normally, the big range and the smaller step increase will increase the recognition accuracy, and also the computational load.

The LS method is very straight and simple. But when the tempo of singing/humming is unstable, the recognition rate will decline because of the global scaling.

3 Segment-Based Approaches

Our approach is to separate the humming/singing into several segments, and find out the corresponding matching segments in midis in database by applying LS on each segment and calculating the distance. This segment-based approach will help us solve the problems of unstable tempo and puff noise which the traditional LS fails to solve. Therefore, how to determine the segments is the key point of our method.

Before discussing our methods of determining segments, we will make an overview of our segment-based method of QBSH. In the segment-based method we proposed, firstly, for query singing/humming and songs in database, we select the points with abrupt change of pitch as the segment boundary candidates. Four approaches will be discussed in detail in the following subsections. The same four methods are applied to midi database to determine the candidate segment boundaries. Secondly, from the segment boundary candidates in query singing/humming, four segment boundaries, the first, the last, the highest, and the second highest boundaries, are chosen and arranged according to their time sequence. Between the segment boundaries, there are, therefore, three segments. Considering the situation that the pitch curves of some songs are quite smooth, which makes the points with abrupt change of pitch may not be many, we only use three segments. In addition, though the distance calculation seems become more accurate if we use more segments, the calculation of matching boundary pairs also becomes more complex. Hence, in our experiments, only four segment boundaries are used. For convenience, when the pitch trend in candidate boundary is upward, we label the segment boundary as positive. On the other hand, if the pitch trend is downward, we label it as negative. Therefore, when the pitch is continuously upward in segment boundary, the boundary is marked as a “positive” boundary from the accumulative result, and when the pitch is continuously downward in segment boundary, the boundary is marked as a “negative” boundary from the accumulative result. We label the four segment boundaries in query and all the candidate segment boundaries in midi database in this way. When we try to find the corresponding matching segment boundaries in song database, it is convenient that we only search from the segment boundary candidates in song database with the same rising (positive) or decreasing (negative) pitch trend. To reduce the computational load, the labeling of segment boundary candidates in midi or song database can be done in advance instead of doing it at query. Thirdly, we need to find out the four corresponding matching segment boundaries pairs in all of the midis. Because the query singing/humming does not necessarily correspond to the very beginning of the midi in database, we need to search from the beginning to the end. However, since the query normally starts from a sentence of a song lyric, we can constrain the search region, for example, the first few numbers of frames from the beginning of a sentence. The corresponding matching segment boundary in midi

is searched from the midi segment boundary candidates with the same rising or decreasing pitch trend. The limitation is the search must be forward to keep the chronological order. For example, when we search the possible corresponding midi point for the second segment boundary, we only search the points following the possible corresponding midi point for the first segment boundary. For all possible matching pairs, we calculate the distance of three segments. Inside segment, linear scaling is applied to stretch or compress the length. The scaling ratio is simply the ratio of segment length, so the query segment is scaled to the same length with the midi segment. Since the expansion or compression of segment bring bias in calculating distance, distance normalization, which normalizes the distance by the length of the pitch vector of the segment is used. Besides, to compromise the effect of key shifting, key transposition, which shifts the pitch to the same mean as that of midi song in database, is adopted. The segment pair with the shortest normalized Euclidean distance is the best match. The Euclidean distance is used to measure the dissimilarity between input query and reference MIDI. The normalized Euclidean distance is as

$$D = \frac{1}{N}\sqrt{\sum_{i=0}^{N-1}(q_i - r_i)^2}\,, \tag{2}$$

Where q_i and r_i are respectively ith query and reference MIDI, and N is the length of the pitch vector of the segment. Fourthly, after compare all the songs in database, the song with the shortest distance is the answer to the query singing/humming.

Our proposed four approaches of the first step, selecting segment boundary candidates, are introduced in the following subsections in detail. We call them "Cross n Semitones", "Moving Average", "Composite Moving Average", and "Combination of Cross n Semitones and Composite Moving Average".

3.1 Cross n Semitones

"Cross n Semitones" selects the points with pitch variation over n semitones within m frames as possible segment boundary candidates. We use "Cross n Semitones" to determine the possible segment boundary candidates because we hope the boundary we find is the place with abrupt pitch change, or note change, not just pitch unstable drift. The n is a threshold. If n is large, we get fewer candidate points. If n is small, we get more candidate points, but too small variation means they may be just caused by pitch drift phenomenon instead of pitch trend, and such selection of n may cause mistake. For convenience, when n equals 3, it is called "Cross 3 Semitones".

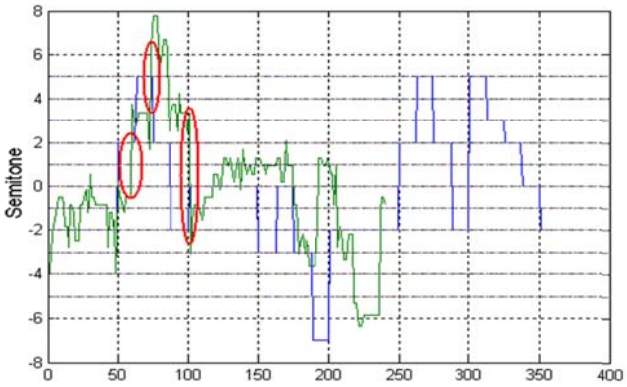

Fig. 1. The pitch curve of humming (green line) and midi (blue stepwise line), and the red circle indicating the place with pitch variation over 3 semitones within 5 frames.

Figure 1 shows the pitch curve of humming (green line) and midi (blue stepwise line), and the red circle indicates the place with pitch variation over 3 semitones within 5 frames. For all the Figures 1-10, the horizontal unit is frame.

3.2 Moving Average

Moving Average (MA), also called rolling average, running average, moving mean or rolling mean, is a calculation method to analyze data points by getting series of averages of different data subsets filtering from sliding windows moving through the full data set. This technique is commonly used to smooth out short-term fluctuations and highlight longer-term trends. The choice of window size will have impact on the degree of smoothing. Longer window creates smoother effect. Mathematically, it can also be considered having the same effect of low-pass filter. It is a method widely used in technical analysis of time series data, like stock or employment. We will apply this technique on finding the possible segment boundaries.

There are drifts in the pitch of query singing/humming. We found out that applying the technique of MA on pitch curve can smooth out short-term pitch fluctuations and highlight the long-term pitch trend, so the interference in the process of selecting the segment boundary candidates can be reduced. The formula for the Moving Average is as below:

$$y[i] = \frac{1}{w}\sum_{j=0}^{w-1} x[i+j]\,. \tag{3}$$

In this equation, x is the input pitch, y is the output smoothed pitch after MA, and w is the number of

points in window, called window length or window size. This equation only uses points on one side. When the window length of moving average is *w*, we call it *w*MA. For example, 20MA means moving average with window length 20. Figure 2 shows the original pitch curves of input query (lower line) and midi (upper stepwise line) in database, and their corresponding smoothed pitch curves after applying 30MA. From Figure 2, it is clear to see that after applying MA, the pitch curve of query and midi becomes more similar than the original ones. Figure 3 is the comparison of input pitch and smoothed pitch curves after applying 10MA, 20MA, and 30MA. From Figure 3, it is easy to see that the greater the *w* is, the smoother the pitch curve is. Smooth curve will reduce the influence of pitch drift, but over smooth curve will lose the original pitch features, resulting a lot of songs look similar. On the other hand, if *w* is small, the resulting pitch curve is less smooth, but keeps more original pitch features and is more vulnerable to the influence of pitch drift. If the pitch trend is upward, for convenience, we label the point as positive, and if the pitch trend is downward, we label the point as negative. The positive or negative value will be accumulated, so the point with consecutive upward (downward) pitch frames will be labeled as positive (negative). If the absolute value of pitch trend (upward or downward) exceeds a threshold *t* we set, we mark it as a possible segment boundary candidates. The same procedure is done on songs in database to find the possible segment boundary candidates. Note that MA only applied in finding possible segment boundary candidates. When calculating the distance, the original pitch is used.

3.3 Composite Moving Average

In MA, the longer the window length *w* is, the smoother the pitch curve is. However, it loses the original pitch features if *w* is too big and the curve is too smooth. On the other hand, if the *w* is small, it is less smooth, and it will keep more original pitch features. But the pitch drift will become interference when we try to locate the point with pitch abrupt change. We also found out that curves after applying MA with different *w* often overlap at points with large dynamic pitch variation, as shown in Figure 4, which presents the original input pitch curve, and the curve after passing 6MA, 13MA, and 20MA. With the combined advantages of small *w* and large *w*, the intersection points can be considered as possible segment boundary candidates. We call this method "Composite Moving Average". The number of MA curves considered is fixed as 3 in our experiment. The descending gap between the window size is *g*. We express it as *w*MA(*g*). For example, 20MA(5) means composite moving average with window length starting from 20 and with descending gap 5, that is, composite of 20MA, 15MA, and 10MA.

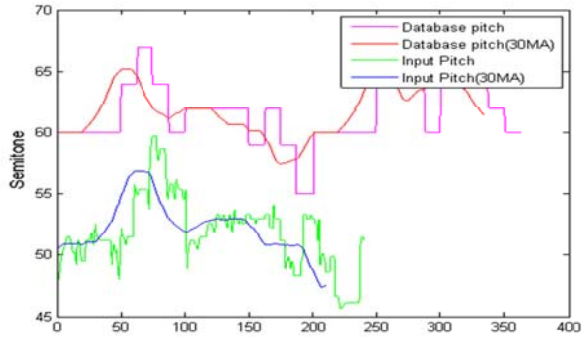

Fig. 2. The original pitch curves of input query (lower line) and midi (upper stepwise line) in database, and their corresponding smoothed pitch curves after applying 30MA.

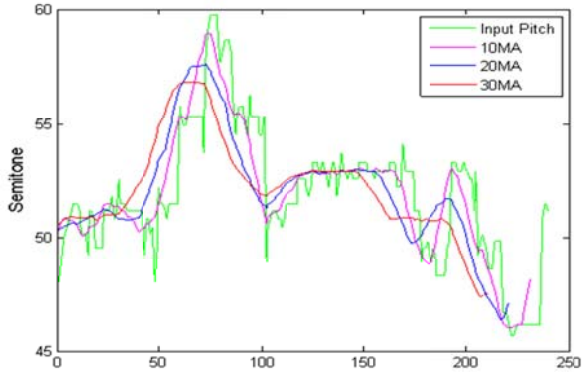

Fig. 3. The comparison of input pitch and smoothed pitch curves after applying 10MA, 20MA, and 30MA.

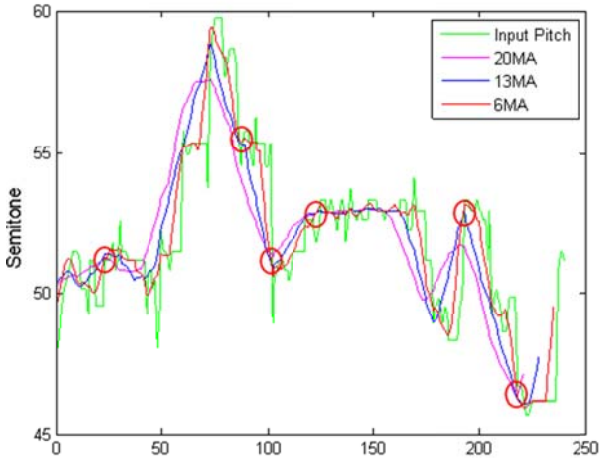

Fig. 4. Original pitch curve, and the curve after passing 6MA, 13MA, and 20MA.

3.4 Combination of Cross n Semitones and Composite Moving Average

This method is a combination of "Cross *n* Semitones" and "Composite Moving Average". If a point is selected by both "Cross *n* Semitones" and "Composite Moving Average", then it is selected as a segment boundary candidate.

4 Experiments

4.1 Database

Two databases are used in our experiment. One is 2009 MIR-QbSH corpus including 35 singers, totally 829 songs. The recording is in wave format with 8 kHz sampling rate, 8 bits per sample, and mono channel. Pitch files are also included. The frame size for pitch is 256 samples. The other database is called Mono-Midi including 342 midi songs, which is the corpus we collected from Internet. The relationship of MIDI note *p* and pitch frequency *f* is

$$p = 69 + 12 \times \log_2\left(\frac{f}{440HZ}\right). \quad (4)$$

4.2 Experimental results

In this section, we will observe the experimental results of our proposed methods and discuss their performance. The recognition results of using different parameters or threshold will also be shown and compared.

First, we observe the impact of choosing different *n* value in "Cross *n* Semitones". Table 1 Shows the comparison of the top 1 and top 10 recognition rates of "Cross *n* Semitones" with n equals 1 to 5 when m equals 5. That is, pitch variation is over *n* semitones within 5 frames. We can see when n equals 3, the recognition rate are best both for top1 and top 10 results. When n is big, the option of possible segment boundaries becomes less and makes the recognition rate decline.

In evaluating MA, we tried 10MA, 20MA, and 30MA with different pitch trend threshold *t*. If the absolute value of pitch trend (upward or downward) exceeds a threshold *t* we set, we mark it as a possible segment boundary candidates. The top1 and top 10 recognition rates with threshold *t* equals 3, 5, 10, and 15, are shown in Table 2. Basically, the recognition rate of 20MA is better in most situations of *t*. Using 30MA makes the curve too smooth to keep the original characteristics, which causes the recognition rate decline. The performance of the threshold 10 is the best. Raising the threshold to 15 makes the recognition rate decline, because the options of possible segment boundaries is not enough.

Table 1. Comparison of the top 1 and top 10 recognition rates of "Cross *n* Semitones" with n equals 1 to 5 when m equals 5.

n	1	2	3	4	5
Top 1	28.5%	47.5%	52.2%	26.8%	6.6%
Top 10	39.9%	57.4%	60.3%	33.2%	13.4%

Table 2. Comparison of the top 1 and top 10 recognition rates of 10MA, 20MA, and 30MA with pitch trend threshold *t* equals 3, 5, 10, and 15.

t		10MA	20MA	30MA
3	Top 1	34.1%	38.1%	35.8%
	Top 10	43.5%	48.7%	47.5%
5	Top 1	43.3%	46.4%	41.5%
	Top 10	54.8%	58.9%	52.8%
10	Top 1	56.0%	55.4%	44.3%
	Top 10	70.1%	70.1%	55.0%
15	Top 1	38.4%	54.2%	43.7%
	Top 10	52.6%	69.6%	54.9%

Table 3. Comparison of the top 1 and top 10 recognition rate of 15MA(3), 15MA(5), 20MA(3), and 20MA(5) with pitch trend threshold *t* equals 10, 20, and 35.

t		15MA(3)	15MA(5)	20MA(3)	20MA(5)
10	Top 1	48.0%	48.3%	51.7%	50.4%
	Top 10	64.3%	63.4%	67.7%	66.5%
20	Top 1	49.8%	51.1%	50.4%	51.7%
	Top 10	68.3%	67.2%	69.7%	69%
35	Top 1	51.7%	51.7%	52%	52.4%
	Top 10	68.3%	68.2%	70.4%	70.6%

Table 4. Comparison of the top 1 and top 10 recognition rate of "Cross n Semitones with *m*=5" and "20MA(5) with *t*=35" with different *n*.

n	1	2	3
Top 1	47.9%	49.6%	56.2%
Top 10	61.4%	62.5%	67.8%

Table 5. Comparion of the Recognition Rate of Our Four Segment-Based Approaches.

	A	B	C	D
Top 1	52.2%	55.4%	52.4%	56.2%
Top 10	60.3%	70.1%	70.6%	67.8%

Then, we compared the top1 and top 10 recognition rate of "Composite Moving Average" - 15MA(3), 15MA(5), 20MA(3), and 20MA(5) with pitch trend threshold *t* equals 10, 20, and 35 as in Table 3. The best result is 20MA(5) with pitch trend threshold 35.

At last, we combined "Cross n Semitones with *m*=5" and "20MA(5) with *t*=35" and compared the recognition rate of different *n*. Table 4 shows the comparison of the top 1 and top 10 recognition rates

of "Cross n Semitones with *m*=5" and "20MA(5) with *t*=35" with different *n*. The best result goes to "Combination of Cross 3 Semitones with *m*=5 and 20MA(5) with *t*=35". Compared with the result in Table 1, combined with "20MA(5) with *t*=35", the top 1 recognition rate of "Cross 3 Semitones with *m*=5" raised from 52.2% to 56.2%, and the top 10 rate raised from 60.3% to 67.8%.

We conclude and compare the top 1 and top 10 recognition rates of our four segment-based methods as shown in Table 5. A, B, C, D respectively indicate our segment-based methods using "Cross 3 Semitones with *m*=5", "20MA with *t*=10", "20MA(5) with *t*=35", and "Combination of Cross 3 Semitones with *m*=5 and 20MA(5) with *t*=35". The top 10 recognition rate of "20MA(5) with *t*=35" can reach 70.6%.

Besides the recognition rate, we want to explore two phenomena; one is puff noise, which usually appears at the beginning of singing when singer put too much air into the microphone, or the other noise happened to get into the microphone, and the other is unstable tempo, which is also very common especially for amateurs. Two examples are provided and observed. The first example is a query singing with puff noise at the beginning. As shown in Figure 5, the (lower) blue line is the singing pitch curve and the (upper) green line is the corresponding midi pitch curve. It is easy to see the puff noise at the beginning of the blue singing pitch curve. After applying our segment-based methods, the singing and midi pitch curves after segment-based scaling are shown in Figure 6. For comparison, the singing and midi pitch curves after applying traditional LS are also shown in Figure 7. Since our methods are segment-based, they are allowed to jump to the most matching segment boundary and exclude the noisy segment. Therefore, puff noise has less impact on the recognition results of our methods than traditional LS. For this example, the correct midi ranks 1, 1, 1, and 1 in experimental results by using our methods A, B, C, and D respectively. It ranks 154 and 139 by using traditional LS and DTW.

Another example is a query singing with inconsistent tempo. As shown in Figure 8, the blue line is the singing pitch curve and the green (stepwise) line is the corresponding midi pitch curve. The singing and midi pitch curves after applying traditional LS are shown in Figure 9. If we align the two curves at the position indicated by (middle) black arrow, we can see the inconsistent tempo causes shift at the position indicated by (left and right) red arrows. The singing and midi pitch curves after segment-based scaling by applying our segment-based methods are shown in Figure 10. The

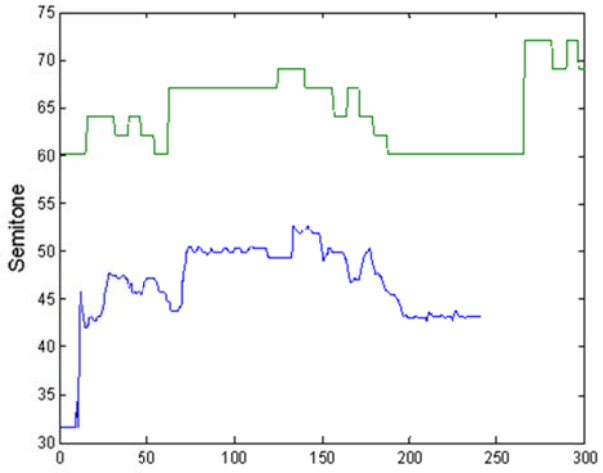

Fig. 5. A singing example with puff noise at the beginning. The (lower) blue line is the singing pitch curve and the (upper) green line is the corresponding midi pitch curve.

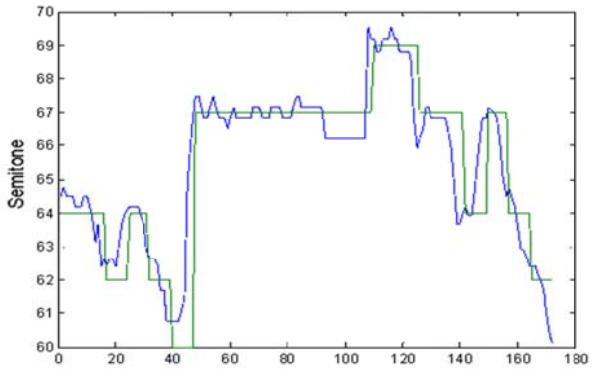

Fig. 6. The singing and midi pitch curves after segment-based scaling. The blue line is the scaled singing pitch curve and the green (stepwise) line is the corresponding midi pitch curve.

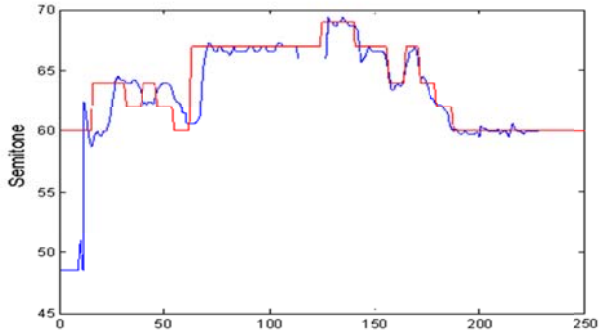

Fig. 7. The singing and midi pitch curves after applying traditional LS. The blue line is the scaled singing pitch curve and the red (stepwise) line is the corresponding midi pitch curve.

two pitch curves fits better than using traditional LS. For this example, the correct midi ranks 1, 1, 1, and 1 in experimental results by using our methods A, B, C, and D respectively. It ranks 86 and 1 by using LS and DTW. It also ranks 1 in DTW because DTW is designed for inconsistent tempo, but its computational complexity is high.

Based on the above experimental results, the problems of puff noise and inconsistent tempo, which usually encountered in QBSH using traditional LS, can be solved by proposed segment-based methods. DTW also can solve tempo inconsistence problem, though, it carries more computational load.

5 Conclusions and future works

We propose segment-based approach to solve the problems of puff noise and inconsistent tempo and try to lower the computational complexity of traditional LS in Query-By-Singing/Humming system for musical retrieval. We used four methods to select segment boundary candidates, including "Cross *n* Semitones", "Moving Average", "Composite Moving Average", and "Combination of Cross *n* Semitones and Composite Moving Average". Experimental results show that the best top 10 recognition rate can reach 70.6%. In addition, from our examples, it shows that segment-based method can solve the problems of puff noise and inconsistent tempo better than traditional global LS. The segment-based methods also reduce the computation complexity by jumping to the possible segment boundary and using the segment ratio as scaling ratio, instead of comparing frame by frame from the very beginning and trying different scaling ratio in traditional LS method. The labeling of segment boundary candidates in midi or song database can be done in advance instead of doing it at query to save time.

Because of the segment nature, segment-based method has better potential to conquer inconsistent tempo problem than global and uniform LS if the segment boundaries is accurate. However, since our method is segment-based and the decision of segment is based on the abrupt change of pitch curve, such approach may encounter problems in music without clear ups or downs. Because in such case, the pitch drift phenomenon may cause impact on selecting corresponding segment boundary pairs and fail the recognition. We observed that when the pitch variation is less than two semitones, it might become an issue. Therefore, the future work that we want to focus on is to improve the segment accuracy

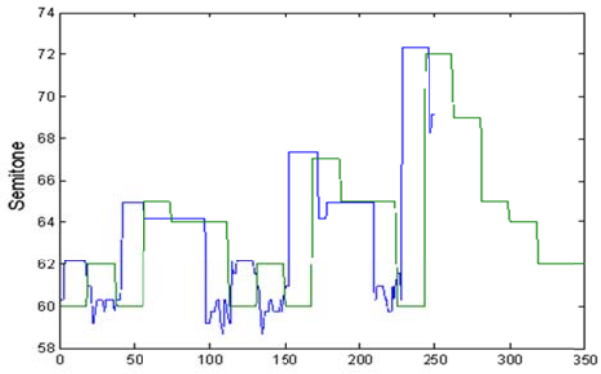

Fig. 8. A singing example with inconsistent tempo. The blue line is the singing pitch curve and the green (stepwise) line is the corresponding midi pitch curve.

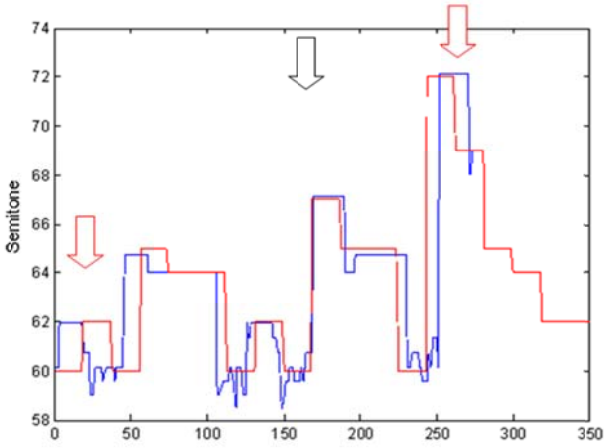

Fig. 9. The singing and midi pitch curves after applying traditional LS. The blue line is the scaled singing pitch curve and the red (stepwise) line is the corresponding midi pitch curve.

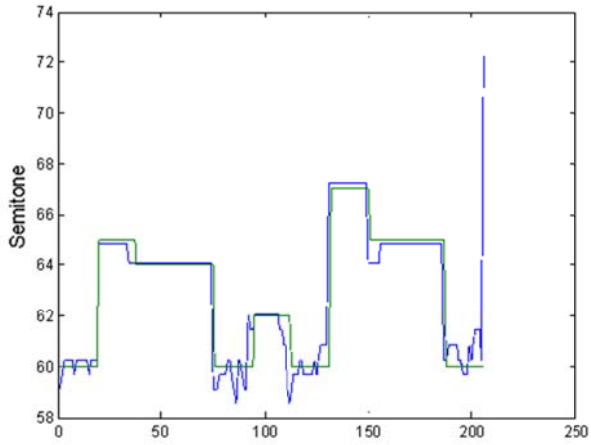

Fig. 10. The singing and midi pitch curves after segment-based scaling. The blue line is the scaled singing pitch curve and the green (stepwise) line is the corresponding midi pitch curve.

by considering more features in addition to pitch variation.

References:

[1] Q. Wang, Z. Guo, G. Liu, C. Li, and J. Guo, Local Alignment for Query By Humming, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, BC, 26-31 May 2013, pp. 3711 – 3715.

[2] C-C. Wang, C.-H. Chen, C.-Y. Kuo, L.-T. Chiu and J.-S. R. Jang, Accelerating Query By Singing/Humming on GPU: Optimization for Web Deployment, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, 25-30 March 2012, pp. 477 – 480.

[3] W.-T. Kao, C.-C. Wang, K. K. Chang, J.-S. R. Jang, and W. Liou, A Two-stage Query by Singing/Humming System on GPU, *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Kaohsiung, Oct. 29-Nov. 1 2013, pp. 1 – 6.

[4] W. Cao, D. Jiang, J. Hou, Y. Qin, T. F. Zheng, and Y. Liu, A Phrase-Level Piecewie Linear Scaling Algorithm for Melody Match in Query-By-Humming Systems, *IEEE International Conference on Multimedia and Expo (ICME)*, New York, NY, June 28 2009-July 3 2009, pp. 942 – 945.

[5] Gi. P. Nam and K. R. Park, Fast Query-by-Singing/Humming System That Combines Linear Scaling and Quantized Dynamic Time Warping Algorithm, *International Journal of Distributed Sensor Networks*, Vol. 2015, Article ID 176091, pp. 1 – 10.

[6] S.-Y. Cheng, An Evaluation of Query By Singing/Humming Based on Various Pitch Tracking Methods, *Master Thesis*, National Tsing Hua University, Taiwan, 2008.

[7] H.-Y. Zhuang, Early Screening of Inappropriate Inputs for QBSH Systems, *Master Thesis*, National Tsing Hua University, Taiwan, 2009.

[8] N.-H. Liu, Effective Results Ranking for Mobile Query by Singing/Humming Using a Hybrid Recommendation Mechanism, *IEEE Transactions on Multimedia*, Vol. 16, No. 5, August 2014, pp.1407-1420.

[9] C.-J. Song, H. Park, C.-M. Yang, S.-J. Jang, and S.-P. Lee, Implementation of a Practical Query-by-Singing/Humming(QbSH) System and Its Commercial Applications, *IEEE International Conference on consumer Electronics (ICCE)* 2013, pp.102-103.

[10] J.-S. R. Jang and Y.-S. Jang, On the Implementation of Melody Recognition on 8-bit and 16-bit Microcontrollers, *ICICS-PCM*, 15-18 December, 2003, Singapore, pp. 704-708.

[11] Z. Guo, Q. Wang, G. Liu, J. Guo, and Y. Lu, A Music Retrieval System Using Melody and Lyric, *IEEE International Conference on Multimedia and Expo Workshops*, 2012, pp.343-348.

[12] W. Wei, L. Z. Xu, and Y. Dong, A Cross-Dimension Indexing Structure for Query-by-S inging, *6th International Conference on Digital Content, Multimedia Technology and its Applications (IDC)*, 16-18 Aug. 2010, pp. 178-182.

[13] C.-C. Wang and J.-S. R. Jang, Improving Query-by-Singing/Humming by Combining Melody and Lyric Information, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 23, No. 4, April 2015, pp. 798-806.

[14] M. Antonelli, A. Rizzi, and G. del Vescovo, A Query by Humming System for Music Information Retrieval, *10th International Conference on Intelligent Systems Design and Applications*, 2010, pp.586-591.

[15] L. Gao and Y. Wu, A System for Melody Extraction from Various Humming Inputs, *IEEE International Symposium on Signal Processing and Information Technology*, 2006, pp. 680-684.

[16] B.-J. Han, S. Rho, and E. Hwang, An Efficient Voice Transcription Scheme for Music Retrieval, *International Conference on Multimedia and Ubiquitous Engineering(MUE)*, 26-28 April 2007.

[17] N. H. Adams, M. A. Bartsch, and G. H. Wakefield, Coding of Sung Queries for Music Information Retrieval, *IEEE Workshap on Applications of Signal Processing to Audio and Acoustics*, October 19-22. 2003, New Paltz, NY, pp. 139-142.

[18] N. H. Adams, M. A. Bartsch, and G. H.Wakefield, Note Segmentation and Quantization for Music Information Retrieval, *IEEE Transactions on Audio, Speech, and*

Language Processing, Vol. 14, No. 1, January 2006, pp. 131-141.

[19] T. Liu, X. Huang, L. Yang, and P. Zhang, Query by Humming: Comparing Voices to Voices, *International Conference on Management and Service Science (MASS)*, 20-22 Sept. 2009.

[20] L. Wang, S. Huang, S. Hu, J. Liang, and B. Xu, An Effective and Efficient Method for Query by Humming System Based on Multi-Similarity Measurement Fusion, *ICALIP*, 2008, pp. 471-475.

[21] K. Kim, K. R. Park, S.-J. Park, S.-P. Lee, and M. Y. Kim, Robust Query-by-Singing/Humming System against Background Noise Environments, *IEEE Transactions on Consumer Electronics*, Vol. 57, No. 2, May 2011, pp. 720-725.

[22] D. Jang, S.-J. Jang, and S.-P. Lee, Test of pitch extraction algorithms for query-by-singing/humming system, *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 27-29 June 2012.

[23] E. Pollashi, A Pitch Tracking System Dedicated to Process Singing Voice for Music Retrieval, *IEEE International Conference on Multimedia and Expo (ICME)*, 26-29 Aug. 2002, pp. 341-344.

[24] C. Francu, C. G. Nevill-Manning, Distance Metrics and Indexing Strategies for a Digital Library of Popular Music, *IEEE International Conference on Multimedia and Expo (ICME)*, 30 July-2 Aug. 2000, pp. 889-892.

[25] J. N. Milner and D. F. Hsu, Music Retrieval by Singing and Humming Using Information Fusion, *Proc. 12th IEEE Int. Conf. on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, 2013, pp. 332-338.

[26] H.-H. Shih, S. S. Narayanan, and C.-C. J. Kuo, An HMM-based Approach to Humming Transcription, *IEEE International Conference on Multimedia and Expo (ICME)*, 26-29 Aug. 2002, pp. 337-340.

[27] B. Schuller, G. Rigoll, and M. Lung, HMM-based Music Retrieval Using Stereophonic Feature Inofrmation and Framelength Adaptation, *ICME*, 2003, pp. II 713-716.

[28] M. Ryynanen and A. Klapuri, Query by Humming of MIDI and Audio Using Locality Sensitive Hashing, *ICASSP*, 2008, pp.2249-2252.

[29] C. Parker, A Tree-Based Method For Fast Melodic Retrieval, *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries (JCDL'04)*, pp. 254-255.

[30] W.-H. Tsai and Y.-M. Tu, An Efficient Query-by-Singing/Humming System Based on Fast Fourier Transforms of Note Sequences, *IEEE International Conference on Multimedia and Expo*, 2012, pp. 521-525.

[31] C.-C. Wang, C.-H. Chen, C.-Y. Kuo and J.-S. R. Jang, Improving Query by Singing/Humming Systems over GPUs, *41st International Conference on Parallel Processing Workshops*, 2012, pp. 561-567.

[32] L. Guo, X. He, Y. Zhang, and Y. Lu, Content-based Retrieval of Polyphonic Music Objects Using Pitch Contour, *ICASSP*, 2008, pp. 2205-2208.

[33] J.-S. Leu, C. Changfan, K.-W. Su, and C.-F. Chen, Design and Implementation of Music Information Retrieval And Gathering Engine (MIRAGE), *10th International Symposium on Pervasive Systems, Algorithms, and Networks*, 2009, pp. 498-501.

[34] B. Stasiak, Follow That Tune – Dynamic Time Warping Refinement for Query by Humming, *New Trends in Audio and Video/Signal Processing Algorithms, Architectures, Arrangements and Applications (NTAV/SPA)*, 27-29 September, 2012, Todz, Poland, pp. 109-113.

[35] D. Jang, C.-J. Song, S. Shin, S.-J. Park, S.-J. Jang, and S.-P. Lee, Implementation of a Matching Engine for a Practical Query-By-Singing/Humming System, *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 14-17 Dec. 2011, pp. 258-263.

[36] C.-J. Song, H. Park, C.-M. Yang, S.-J. Jang, and S.-P. Lee, Implementation of a Practical Query-by-Singing/Humming (QbSH) System and Its Commercial Applications, *IEEE Transactions on Consumer Electronics*, Vol. 59, No. 2, May 2013, pp. 407-414.

[37] A. N. Myna, V. Chaitra, and K. S. Smitha, Melody Information Retrieval System using Dynamic Time Warping, *World Congress on Computer Science and Information Engineering*, 2009, pp. 266-270.

[38] R. A. Putri and D. P. Lestari, Music Information Retrieval using Query-by-Humming based on the Dynamic Time Warping, *The 5th International Conference on Electrical Engineering and Informatics*, August 10-11, 2015, Bali, Indonesia, pp. 65-70.

[39] S. Park, and K. Chung, Query by Singing/Humming (QbSH) System for Polyphonic Music Retrieval, *IEEE International Conference on Consumer Electronics (ICCE)*, 2012, pp. 245-246.

[40] Y. Kim and C. H. Park, Query by Humming by Using Scaled Dynamic Time Warping, *International Conference on Signal-Image Technology & Internet-Based Systems*, 2013, pp. 1-5.