Query by Singing and Humming

CHIAO-WEI LIN

Abstract

Music retrieval techniques have been developed in recent years since signals have been digitalized. Typically we search a song by its name or the singer's name if we already know the information. What about if we remember none of the names of the song or singer but know how to sing? Query by singing and humming (QBSH) is an automatic system to identify a song hummed or singed in content-based methods. The basic idea of QBSH system and some techniques to improve the performance are introduced in this paper.

Contents

I. Introduction

Music is part of the lives of people all around the world and it exists a numerous of

styles and forms. Most of the signals have been digitalized nowadays, music is no exception, and it leads to auto processing and analyzing by computers.

The main procedure of a conventional QBSH system is as follows: (1) apply onset detection to identify the notes of the input singing or humming signal (2) extract the pitch of each of the identified notes (3) compare the pitch sequence with database to find the most likely song. The details are introduced in the report.

Some methods of onsets detection are first be introduced in the next section. In section III, some pitch extracting techniques are described. The melody matching methods are introduced in section IV. Last parts of this paper are conclusions and references of this paper.

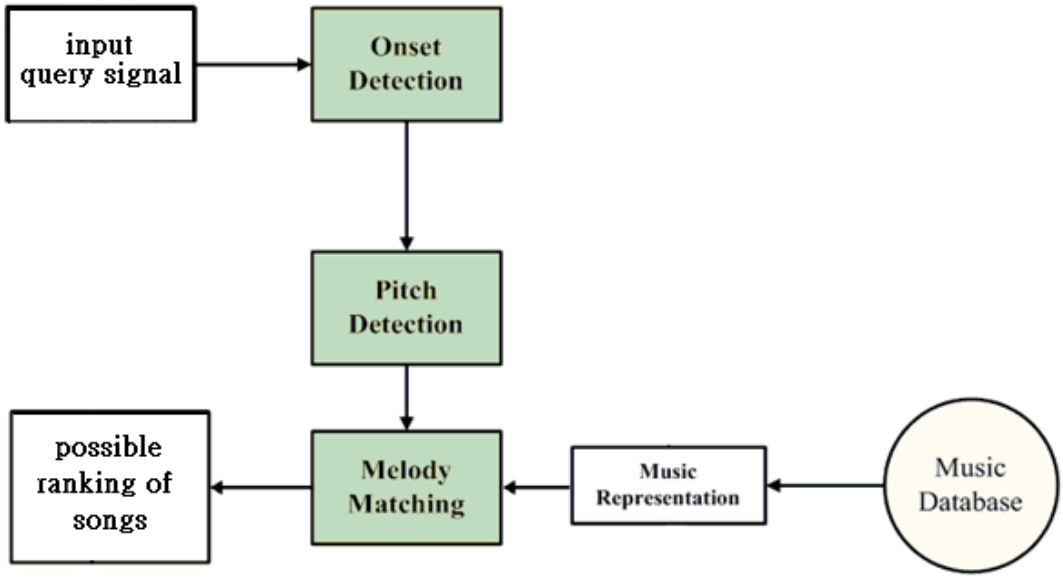

Figure 1 The system diagram of a typical QBSH system

II. ONSET DETECTION

Figure 2 from [1] shows an ideal case of an isolated note. Onset refers to the beginning of a sound or music note. The objective of onset detection is to find onsets in a piece of given music. The basic idea is to capture the sudden changes of volume in music signal. Many different methods have been proposed for this task [1-6].

Here is the basic procedure of onset detection algorithms: pre-processing the original audio signal to improve the performance, then a detection function is applied to do peak-picking which are the locations of the onsets. If the detection function has been designed finely, then onsets events will give rise to well-localized identifiable features

in the detection function.

In the following subsections, several onset detection methods are introduced. Magnitude method, short-term energy method, surf method [4] and envelope match filter [3] are described in detail. To show the performance of each method, we use the signal in figure 3, which has 12 onset points, as example.

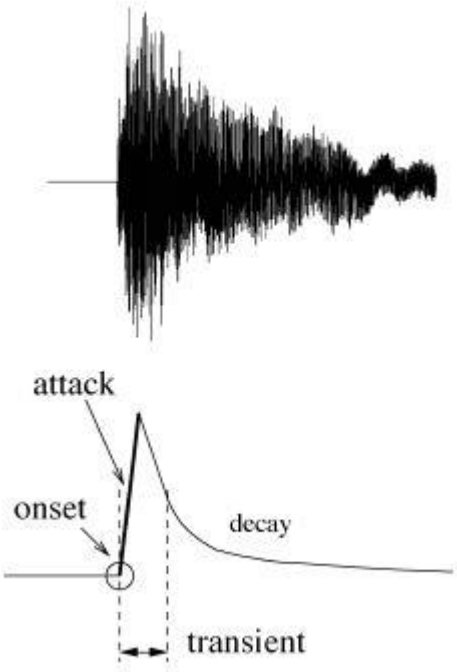

Figure 2 Ideal case of a note.

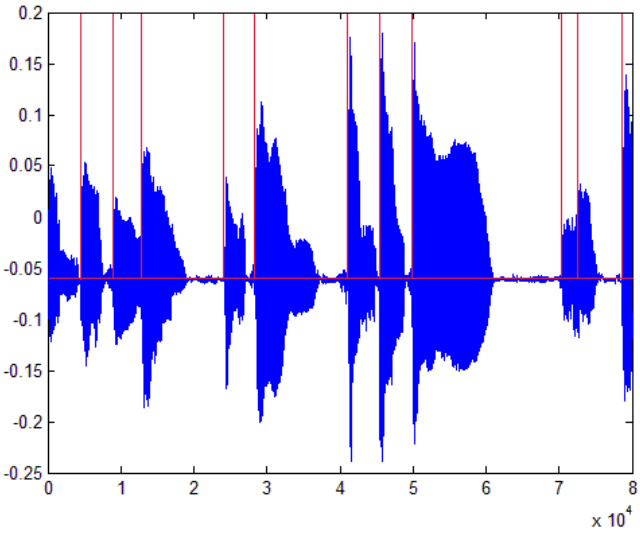

Figure 3 The real onset points denoted in red lines.

A. Magnitude Method

The magnitude method is the most straightforward for people to get in. This method uses volume as the feature to do onset detection. It uses the difference of the envelope of the input signal to detect the possible onset locations. The process is as follows:

(i) $A_k = \max(LPF\{x[n]\}|kn_0 \leq n \leq (k+1)n_0)$, (1)

Where $x[n]$ is the input signal, n_0 is the window size and LPF is a low-pass filter.

(ii) $D_k = A_k - A_{k-1}$ (2)

(iii) If $D_k >$ threshold, kn_0 is recognized as the location of onset.

Step 1 is the function to determine the envelope of input signal, and step 2 gets the difference. If the difference gotten in step 2 over the threshold value, it means that there is a sudden, sufficient energy growth, which is exactly the position of onset.

The result of example signal after applying magnitude method is shown in figure 4. Note that there are 13 onset points, which is over-detected. This method is very simple

but highly effected by the background noise and the chosen threshold value. If the threshold value is too small, then the onsets result would be over-detected. On the other hand, if the chosen threshold value is too large, the results would be under-detected. Also note that if the input signal has loud background noise, then the magnitude of signal may not increase abruptly, thus would not be detected as an onset.

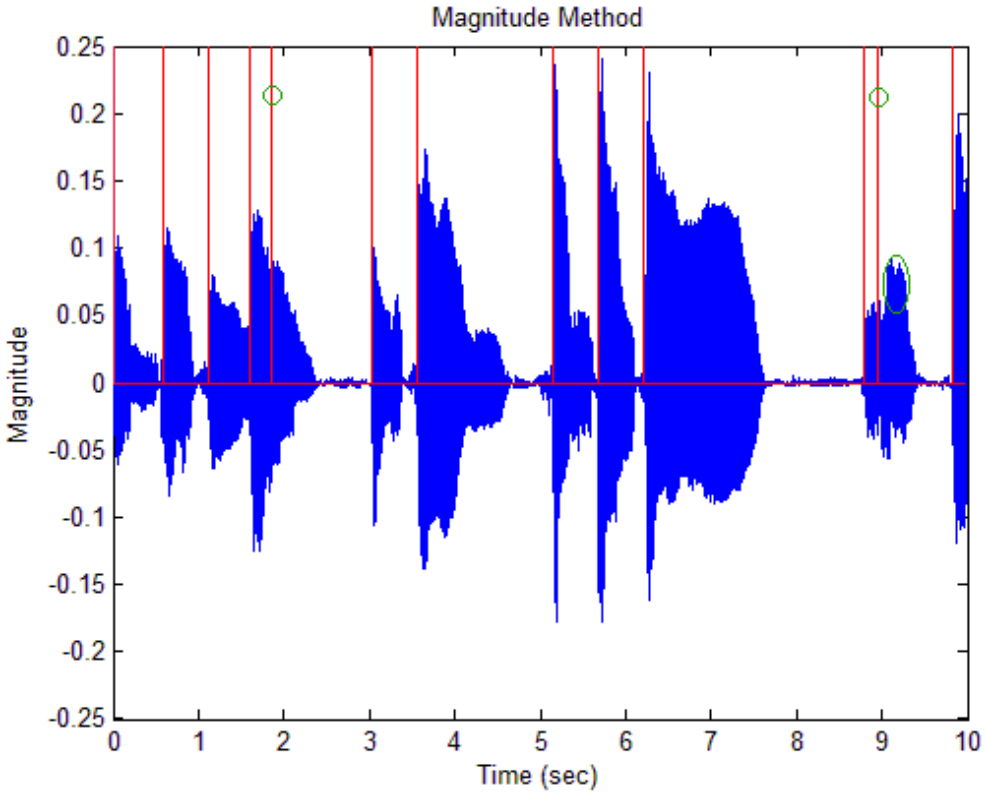

Figure 4 The result of magnitude method.

B. Short-term Energy Method

This approach is also easy to implement. It uses the assumption that there are always silences between consecutive notes. There are two ways to decide the position of onsets. The first one is similar to the magnitude method but uses the energy instead of the envelope as the feature. When onsets happen, the energy difference would over the threshold value. Its process is:

(i) $E_k = \sum_{n=kn_0}^{(k+1)n_0-1} x^2[n]$ (3)

(ii) $D_k = E_k - E_{k-1}$ (4)

(iii) If $D_k >$ threshold, kn_0 is recognized as the location of onset.

The first step is to calculate the total energy in the window with window size equals to n_0. How to choose an appropriate threshold value is the most important issue in this onset detection method. Just as magnitude method, if the chosen value is too small, the

onsets will be over-detected.

The second way to implement this approach is as follows:

(i) $E_k = \sum_{n=kn_0}^{(k+1)n_0-1} x^2[n]$ (3)

(ii) $D_k = \begin{cases} 1, & \text{if } E_k > \text{threshold} \\ 0, & \text{if } E_k \leq \text{threshold} \end{cases}$ (5)

(iii) For each continuous 1-sequence, set the first one as onset and the last one as offset.

The first step implements just as the first way. Note that after step 3, there are only three values: 0, 1 and -1. 1means onset, and -1 means offset. The value of 0 means that there is no obvious change of energy.

Figure 5 is the result after applying short-term energy method to detect onset points. From that, we can see that the result is highly affected by the threshold value. In figure 5(a), there are 14 detected onset point while there are only 10 in figure 5(b).

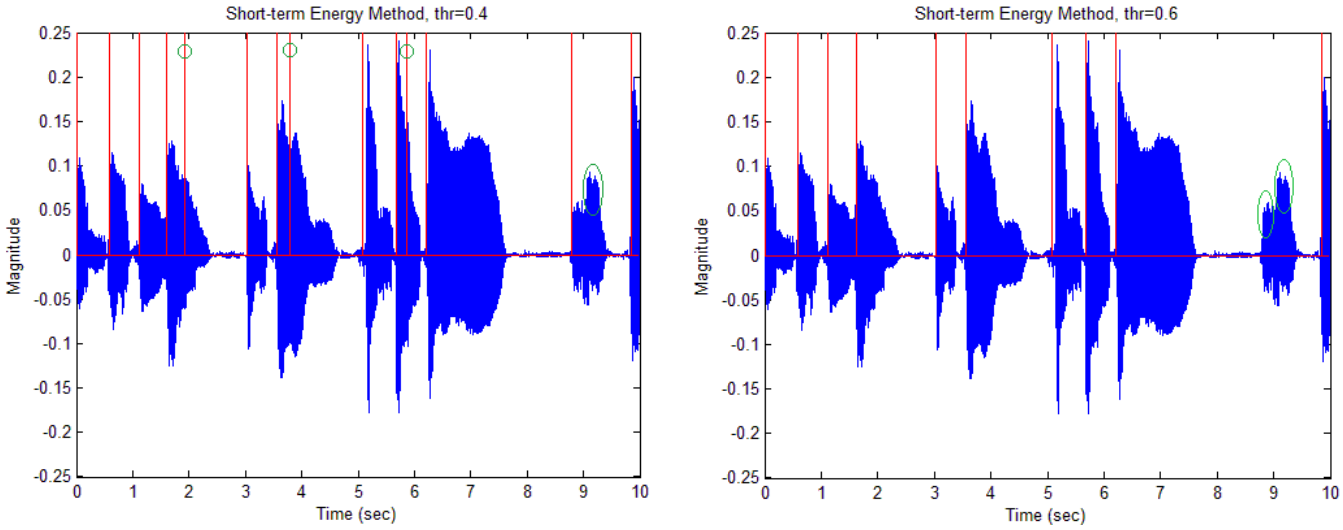

Figure 5 The result of example signal after applying short-term energy method with (a) threshold values equal to 0.4 and (b) 0.6 respectively.

C. Surf Method

The surf method proposed by Pauws [4] uses the slope that is calculated by fitting a second-order polynomial function to detect onsets. The procedure is as follows:

(i) $A_k = \max(x[n] | kn_0 \leq n \leq (k+1)n_0)$, (1)

Where $x[n]$ is the input signal, n_0 is the window size.

(ii) Approximate A_m for m=k-2 ~ k+2 by a second-order polynomial function $p[m] = a_k + b_k(m-k) + c_k(m-k)^2$. The coefficients b_k is the slope of the center (m=0):

$$b_k = \sum_{\tau=-2}^{2} A_{k+\tau}\tau \,/\, \sum_{\tau=-2}^{2} \tau^2. \quad (6)$$

(iii) If b_k > threshold, kn_0 is recognized as the location of onset.

This method is more precise than the magnitude method and the short-term energy method, but needs more computation time. Also, the surf method tends to be over-detected since that when people sings a note, there is a slight off-pitch in the end of the sound. The result after applying surf method is shown in figure 6. There are two over detections and one miss.

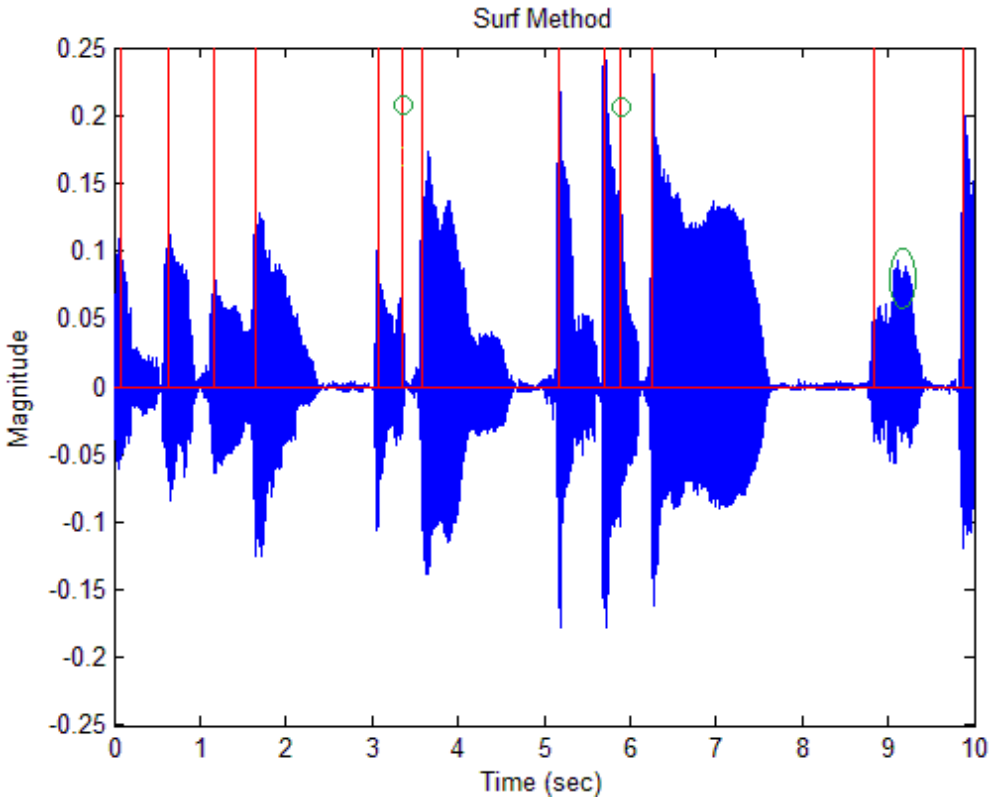

Figure 6 The result of surf method.

D. Envelope Match Filter

For onsets detection, another approach was proposed to enhance the performance [3]. The assumed shape of an attacking signal in Figure 7(b) is obtained by observing the shape of a humming signal as Figure 7(a). From this observation, the match filter $f[n]$ which is the time reversal of Figure 7 (b) is used to find out the onsets. Before applying the match filter, pre-processing like normalization and fractional power are taken.

The process is:

(i) $A_k = \max(x[n] | kn_0 \le n \le (k+1)n_0)$, (1)

Where $x[n]$ is the input signal, n_0 is the window size.

(ii) $B_k = (\frac{A_k}{0.2+0.1*A_k})^{0.7}$ (7)

(iii) $C_k = convolution(B_k, f)$, (8)

Where f is the match filter described above.

(iv) If $C_k >$ threshold, then kn_0 is recognized as the location of onset.

Figure 8 is the result after applying the enveploe match filter on the example signal.

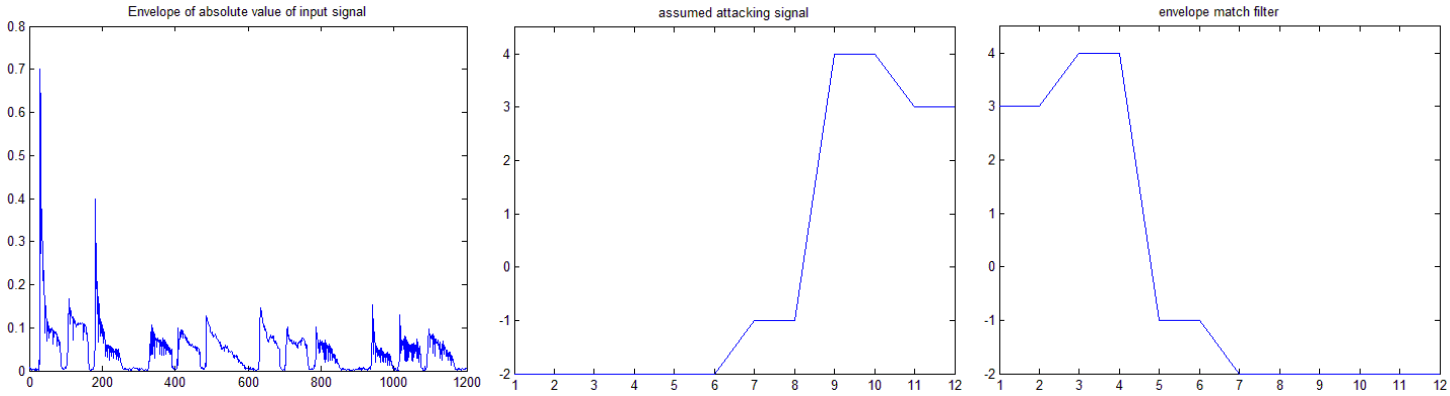

Figure 7 (a) The envelope of a humming signal. (b) Assuming of an attacking signal. (c) The match filter.

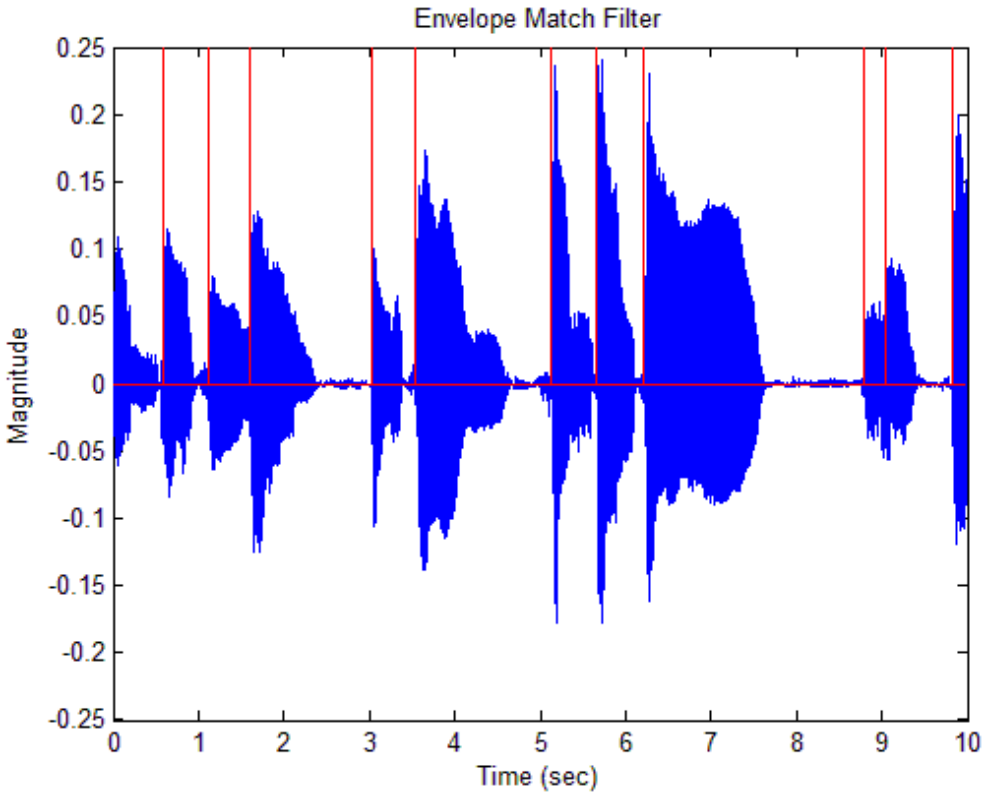

Figure 8 The result of envelope match filter.

III. PITCH EXTRACTION

After the onsets detection, the next thing to do is estimating the fundamental frequency of each note. Pitch is one of the most important and universal feature of

music pieces. There are some existing approaches for computing the fundamental frequency [7-16]. Generally, pitch tracking method can be classified into the time domain and the frequency domain[13]. Time-domain method includes autocorrelation function (ACF) and average magnitude difference function (AMDF). Sub-harmonic summation and harmonic product spectrum (HPS) [7] are some of pitch extraction method in the frequency domain. The Sub-harmonic summation [8] uses the logarithmic frequency to represent the sub-harmonic sum spectrum and produce a virtual pitch. An auditory sensitivity filter is used to fit human perception. The Hilbert-Huang Transform proposed by Huang in 1998 [14] is a pitch tracking method that is robust when fundamental frequency exceeds 600 Hz, but need more computation time. Also, it does not perform well when the input signal has loud background noise. The proposed method is much simpler. ACF, AMDF, HPS and our method are introduced below.

A. Autocorrelation Function

Autocorrelation function (ACF) [15] is particularly useful in estimating hidden periodicities in signal. The function is:

$$\mathrm{ACF}(n) = \frac{1}{N-n}\sum_{k=0}^{N-1-n} x(k)x(k+n) \tag{9}$$

Where *N* is the length of signal *x*, *n* is the time lag value. The value of *n* that maximize ACF(*n*) over a specified range is selected as the pitch period in sample points. If ACF has highest value at *n*=K, then K is the chosen time period of signal, and the fundamental frequency is 1/K.

Figure 9 taken from [13] demonstrate the operation of ACF. To get ACF, we need to shift n for N times, for each time compute the inner product of the overlap parts.

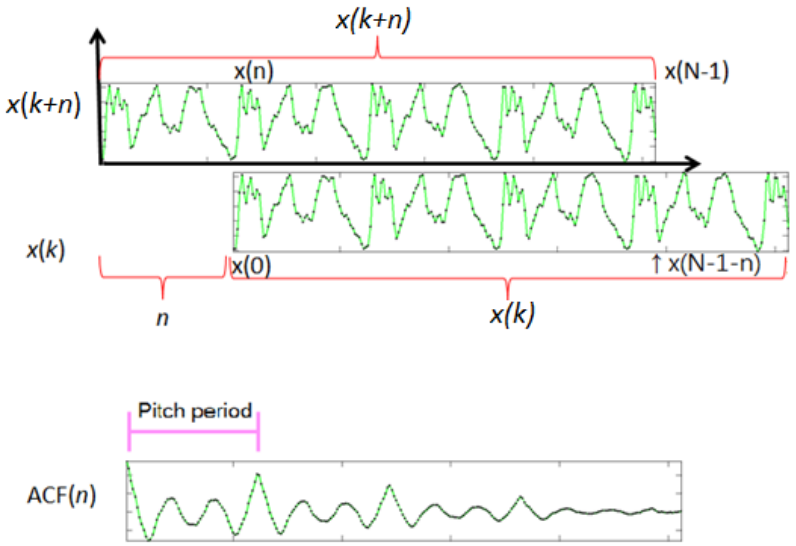

Figure 9 Demonstration of ACF.

B. Average Magnitude Difference Function

The concept of average magnitude difference function (AMDF) [16] is very similar to ACF, except that it uses the distance instead of similarity. The formula is as follows:

$$\mathrm{AMDF(n)} = \frac{1}{N-n}\sum_{k=0}^{N-1-n}|x(k) - x(k+n)| \tag{10}$$

Where N is the length of signal x, and n is the time lag value. As figure10 shows, AMDF counts the sum of difference of overlap regions. The first value of n in AMDF(n) that is approximate to 0 is selected as the pitch period in sample points.

The demonstration is in Figure 10 [13]. Unlike ACF which find the maximum position, the value that minimizes AMDF over a specified range is selected as the pitch period. If the lowest value occurs at n=K, then K is the chosen time period of signal, and the fundamental frequency is 1/K.

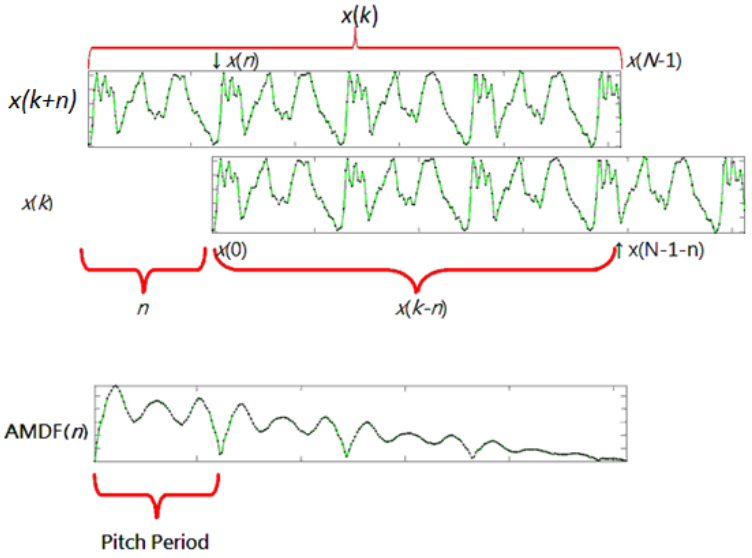

Figure 10 Demonstrate of AMDF [13].

C. Harmonic Product Spectrum

Harmonic product spectrum is a pitch extraction method proposed by MR Schroeder in 1968 [7]. Unlike ACF and AMDF, harmonic product spectrum (HPS) is one of the pitch extraction method in the frequency domain. The schematic diagram is shown in figure 9 [13]. The procedure is as follows:

(i) X=FT$\{x\}$, (11)

Where FT is the Fourier transform and x is signal in the time domain.

(ii)$X_m = downsample(x, m)$, for m = 1~M. (12)

That is, keep only the multiple of m points of X.

(iii)$Y = \sum_{m=1}^{M} X_m$ (13)

(iv) Fundamental frequency f is the frequency that has the largest energy in Y.

The reason that we can use this method to estimate the fundamental frequency is that there are harmonics, which are integer multiples of the fundamental frequency. This method can sum up the energy of harmonic, thus highlight the fundamental frequency.

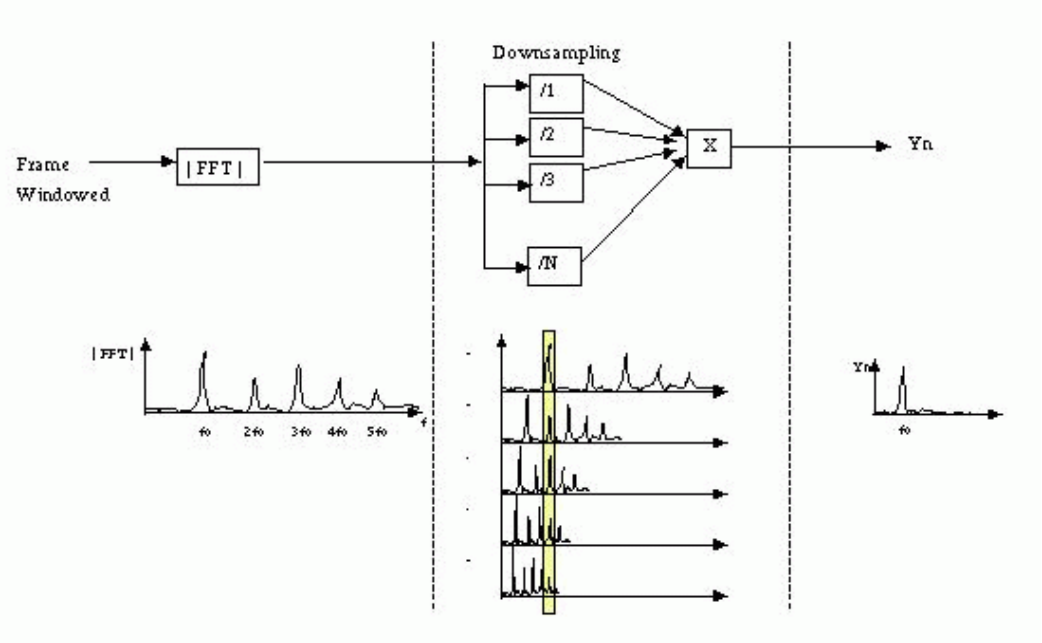

Figure 11 The schematic diagram of harmonic product spectrum[13].

D. Proposed Method

Since the humming signal is always single tone, a much simpler method can be used to detect fundamental frequency. The energy at harmonics is obviously larger than other frequency, thus we can get the fundamental frequency simply by finding the top 3 peaks in the frequency domain and choose the lowest one. The procedure is as follow:

(i) X=FT$\{x\}$, (11)

Where FT is the Fourier transform and x is signal in the time domain.

(ii) Get the top 3 peaks f_1, f_2, f_3.

(iii) Fundamental frequency=min (f_1, f_2, f_3).

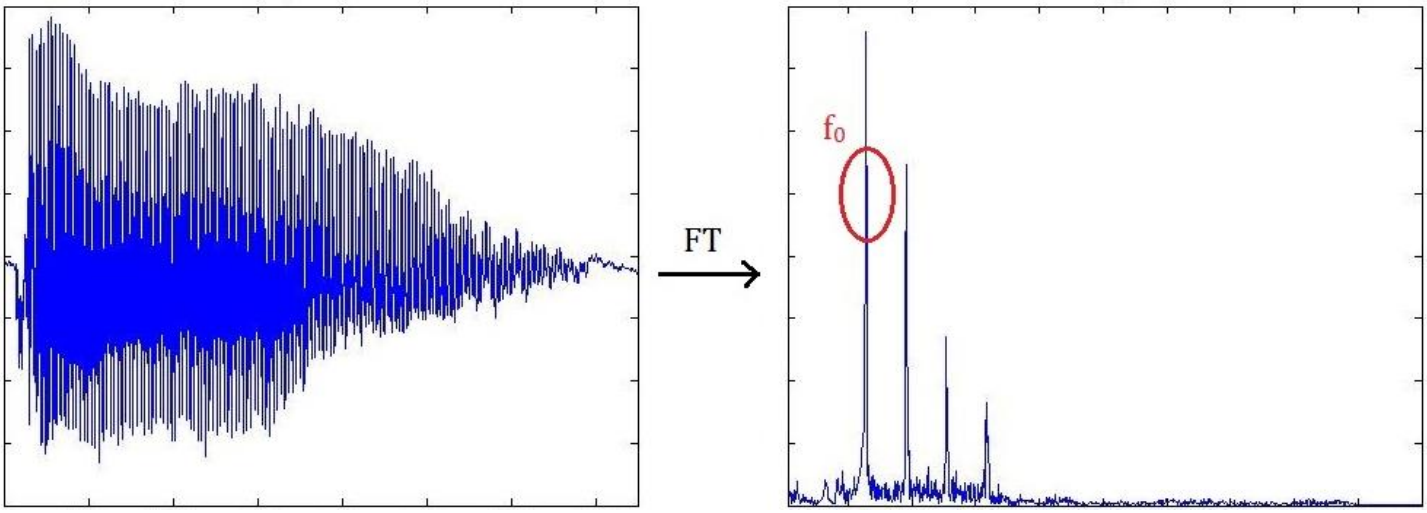

Figure 12 The process of proposed method

IV. MELODY MATCHING

After the fundamental frequency of query are extracted, we transfer the pitch sequence into MIDI number for melody matching. In the melody matching stage, comparing the MIDI sequence with those in the database, any song in database that get higher matching score is the probable matching song. However, there are some situations which might lead to error matching or increasing the matching difficulty like that people might sing at wrong key, sing too many or too few notes or sing from any part of the song. A good matching method should be able to conquer these problems.

Some basic matching methods including dynamic programming, hidden Markov model and linear scaling have been proposed. Linear scaling is a melody matching method proposed in 2001 [17]. This algorithm simply stretches or compressed the query pitch sequence and match it point by point with targets in database. However, if the rhythm of query deviates from the original song too much, this method would lead to a lot of mismatch. Dynamic programming is a method proposed in 1956 to find an optimal solution to multistage problem. In this chapter, the algorithms of melody matching are introduced below.

A. Hidden Markov Model

After pitch estimating, we have got the information of a humming signal and can see each note as a situation. From the reason that the notes are consecutive, we can use pitch sequence to construct a transition model of a piece of music. Markov Model for

melody matching is a probability transition cycle which consists of a series of specific states characterized by *pitch*. Each states has a transition probability to the other states. It represents a process going through a sequence of discrete states. There are three basic elements to form a Markov Model:

(1)A set of states $S = \{s_1, s_2, \dots, s_N\}$, N is the number of states.

(2)A set of transition probabilities *T*, where $t_{i,j}$ in *T* represents the transition probability from state s_i to s_j. The transition probabilities can be formed as an NxN transition matrix *A*.

(3)A initial probability distribution π, where π_i is the probability that the sequence begins in state s_i.

Each song in database has its own Markov Model which is created by the feature of the song itself. An example is illustrated in Figure 13.

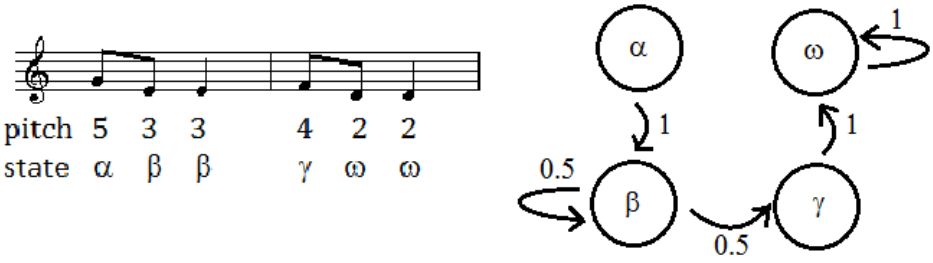

Figure 13 An example of a Markov Model.

Hidden Markov Model (HMM) [18] is an extended version of Markov Model. Unlike Markov Model, each observation is a probability function instead of a one-on-one correspondence of each state, that is, a node is a probability function of states instead of only one state. Thus, there is one more element of an HMM besides S, A and π mentioned above:

(4)*B*, a set of *N* probability functions, each describing the observation probability with respect to a state.

The hidden Markov model for melody matching is described below. In a hidden Markov Model, there is no zero-probability transition exists due to every transition might happens. In this approach, every target in database and the query is an observation sequence $\bar{O} = (o_1, o_2, \dots, o_T)$, each o_i is characterized by *pitch*. First, we construct a hidden Markov model of every song in database. The probability of observation o_i can be estimates by counting the times o_i happen and comparing to the total number of times *s* are encountered:

$$\mathrm{P}(o_i|s) = \frac{count(o_i,s)}{\sum_{j=1}^{|o|} count(o_j,s)} \tag{12}$$

For the observations that do not occur in training data, we need to give them a minimal probability P_m since we could not ensure that they will never occur. The last step of

building a hidden Markov model is to renormalize the transition probability again. Using the example in figure 7 and the assumption that all the possible states are {α, β, γ, ω, ε} to demonstrate HMM, the resulting transition table is shown in Table 1 and Table 2. Here we take $P_m = 0.05$ as example. Table 1 is the result which we give a small probability to those transitions do not observed, and Table 2 is the result of normalization of Table 1.

From / To	α	β	γ	ω	ε
α	0.05	0.05	0.05	0.05	0.05
β	1	0.5	0.05	0.05	0.05
γ	0.05	0.5	0.05	0.05	0.05
ω	0.05	0.05	1	1	0.05
ε	0.05	0.05	0.05	0.05	0.05

Table 1 The result of realigning transition table with $P_m = 0.05$.

From / To	α	β	γ	ω	ε
α	0.0425	0.0434	0.0425	0.0425	0.2
β	0.8333	0.4348	0.0425	0.0425	0.2
γ	0.0425	0.4348	0.0425	0.0425	0.2
ω	0.0425	0.0434	0.8333	0.8333	0.2
ε	0.0425	0.0434	0.0425	0.0425	0.2

Table 2 The result of final HMM.

B. Dynamic Programming

Dynamic programming (DP) [19] proposed by Richard Bellman is a method to find an optimum solution to a multi-stage decision problem. This method has been used in DNA sequence matching for a long time. It can be used to compare a MIDI sequence

with those in database likewise.

Let *Q* and *T* denote the query and target MIDI sequence respectively, while $|Q|$ and $|T|$ as the sequence length. Create a matrix *AlignScore* with$|Q|+1$ rows and$|T|+1$ columns where $AlignScore(i\,j)$ is the score of the best alignment between the initial segment q_1 through q_i of Q and the initial segment t_1 through t_i of T. The boundary conditions are $AlignScore(i,0) = -i$ and $AlignScore(0,\ j) = -j$. The best score is decided by:

$$AlignScore(i,j) = \max\begin{cases} AlignScore(i-1,j-1) + matchScore(q_i,t_j) \\ AlignScore(i-1,j) - 1 \\ AlignScore(i,j-1) - 1 \end{cases} \quad (13)$$

Where the *matchScore* is defined as

$$matchScore(q_i,t_j) = \begin{cases} 2,\ if\ q_i = t_j \\ -2,\ otherwise \end{cases} \quad (14)$$

The $matchScore(q_i,t_j)$ function in the top line means the reward of a match or mismatch respectively, and the -1 in the following two lines represents the skip penalty of *"insertion"* and *"deletion"*. *Insertion* is the situation that there are more elements in one sequence than the other while *deletion* is that there are some missing elements.

See Table 3 for an example. The direction of arrows are the route of tracing back to find the parents used to generate the score in cell. The vertical and horizontal arrows denote a deletion and insertion respectively. As we can see in Table 3, there are four maximum score routes. The maximum score alignments are shown in Table 4, where a dash – is a skip (insertion or deletion).

Target / Query		G	A	B	B
	0	-1	-2	-3	-4
G	-1	2	1	0	-1
D	-2	1	0	-1	-2
A	-3	0	3	2	-1
C	-4	-1	2	1	0
B	-5	-2	1	4	3

Table 3 The alignment matrix with the maximum score alignment.

route	1	2	3	4
Target	G - AB - B	G - A - BB	G - ABB	G - A - BB
Query	GDA - CB	GDAC - B	GDACB	GDAC B -

Table 4 Four maximal-scoring alignments

C. Linear Scaling

Linear scaling is proposed by J. Jang in 2001 [17]. This algorithm is a straightforward melody matching method at frame level. Since this method is frame-based, the rhythm information is included. When humming a song, people might not sing in the same speed as the original song. When human sings without music, the speed is often between 0.5 to 1.5 times of the original one. For this reason, the query pitch sequence is linearly scaled several times to compare with the songs in database.

The algorithm is very simple: it simply stretches or compresses the query pitch sequence and compute the distance to targets in database point by point. It involves some factors: scaling factor, scaling-factor bounds and resolution. The scaling factor is the length ratio between the scaled and the original sequence, and the scaling-factor bounds are the upper and lower bounds. The resolution is the number of scaling factor. For the example in Figure 14 taken from [20], the resolution is 5 and the scaling-factor bounds are 0.5 and 1.5. The next step after stretching or compressing the input sequence is compare all these scaled versions with each song in database, and take the minimum distance as the distance between query and this song. In the example, the distance of the song in database to the query is the distance between the song in database and 1.25 times stretched version of query.

The advantage of this method is that it has low complexity. However, if the rhythm of query deviates from the original song too much, this method would lead to a lot of mismatch. This method also needs well training to capture human's singing habits.

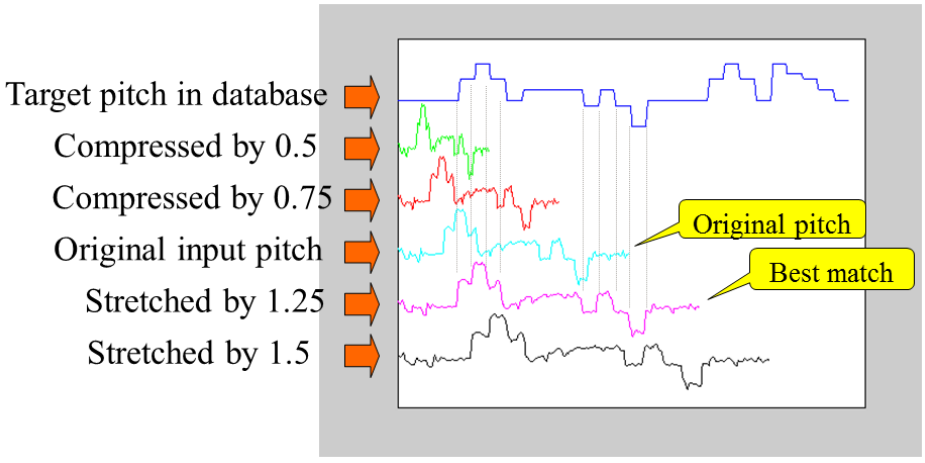

Figure 15 Example of linear scaling with the best scaling factor 1.25.

V. CONCLUSIONS

Query-By-Singing and Humming system makes people search their desired songs by content-based method. In this paper, the QBSH system and some basic algorithms for it were introduced. The first step of QBSH is onsets detection, which was introduced in section II. In section III, the basic idea of pitch tracking was described. We introduced some pitch estimating methods like ACF and HPS in this section. The forth part talked about hidden Markov model and dynamic programming, which are useful for melody matching. These methods are helpful in music signal processing.

VI. REFERENCES

[1] J. P. Bello, L. Daudet, S. Abdallah *et al.*, "A tutorial on onset detection in music signals," *Speech and Audio Processing, IEEE Transactions on,* vol. 13, no. 5, pp. 1035-1047, 2005.

[2] S. Hainsworth, and M. Macleod, "Onset detection in musical audio signals," *Proc. Int. Computer Music Conference,* pp. 163-6, 2003.

[3] J.-J. Ding, C.-J. Tseng, C.-M. Hu *et al.*, "Improved onset detection algorithm based on fractional power envelope match filter," *Signal Processing Conference, 2011 19th European,* pp. 709-713, 2011.

[4] S. Pauws, "CubyHum: a fully operational" query by humming" system," *ISMIR,* pp. 187-196, 2002.

[5] J. P. Bello, and M. Sandler, "Phase-based note onset detection for music signals," *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP'03). 2003 IEEE International Conference on,* pp. V-441-4 vol. 5, 2003.

[6] S. Abdallah, and M. D. Plumbley, "Unsupervised onset detection: a probabilistic approach using ICA and a hidden Markov classifier," *Cambridge Music Processing Colloquium,* 2003

[7] M. R. Schroeder, "Period Histogram and Product Spectrum: New Methods for Fundamental-Frequency Measurement," *The Journal of the Acoustical Society of America,* vol. 43, no. 4, pp. 829-834, 1968.

[8] D. J. Hermes, "Measurement of pitch by subharmonic summation," *The journal of the acoustical society of America,* vol. 83, no. 1, pp. 257-264, 1988.

[9] E. Tsau, N. Cho, and C.-C. J. Kuo, "Fundamental frequency estimation for music signals with modified Hilbert-Huang transform (HHT)," *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. IEEE, pp. 338-341, 2009.

[10] E. Pollastri, "Melody-retrieval based on pitch-tracking and string-matching

methods," *Proc. Colloquium on Musical Informatics, Gorizia*. 1998.

[11] S. Kadambe, and G. F. Boudreaux-Bartels, “Application of the wavelet transform for pitch detection of speech signals,” *IEEE Transactions on Information Theory,* vol. 38, no. 2, pp. 917-924, 1992.

[12] L. Rabiner, M. J. Cheng, A. E. Rosenberg *et al.*, “A comparative performance study of several pitch detection algorithms,” *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 24, no. 5, pp. 399-418, 1976.

[13] J.-S. R. Jang, “Audio signal processing and recognition,” *Information on http://www. cs. nthu. edu. tw/~ jang*, 2011.

[14] N. E. Huang, Z. Shen, S. R. Long *et al.*, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis." *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. vol. 454. no. 1971. The Royal Society, pp. 903-995, 1998.

[15] X.-D. Mei, J. Pan, and S.-h. Sun, "Efficient algorithms for speech pitch estimation." *Intelligent Multimedia, Video and Speech Processing. Proceedings of 2001 International Symposium on*. IEEE, pp. 421-424, 2001.

[16] M. J. Ross, H. L. Shaffer, A. Cohen *et al.*, “Average magnitude difference function pitch extractor,” *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 22, no. 5, pp. 353-362, 1974.

[17] J.-S. R. Jang, H.-R. Lee, and M.-Y. Kao, "Content-based music retrieval using linear scaling and branch-and-bound tree search." *null*. IEEE, p. 74, 2001.

[18] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE,* vol. 77, no. 2, pp. 257-286, 1989.

[19] R. Bellman, “Dynamic programming and Lagrange multipliers,” *Proceedings of the National Academy of Sciences of the United States of America,* vol. 42, no. 10, pp. 767, 1956.

[20] J.-S. R. Jang, and H.-R. Lee, “A general framework of progressive filtering and its application to query by singing/humming,” *Audio, Speech, and Language Processing, IEEE Transactions on,* vol. 16, no. 2, pp. 350-358, 2008.