Applied Mathematics & Information Sciences
An International Journal

A Query-by-Humming System based on Marsyas Framework and GPU Acceleration Algorithms

Li Ruan[1], *Long Wang*[1], *LiMin Xiao*[1], *MingFa Zhu*[1] *and YiLei Wu*[2]

[1]State Key Laboratory of Software Development Environment, Beihang University, 100191 Beijing, China
[2]Beijing Computing Center,100094 Beijing,China

Received: 11 Aug. 2012, Revised: 22 Nov. 2012, Accepted: 2 Dec. 2012
Published online: 1 Feb. 2013

Abstract: Query-by-humming(QBH) is a content-based music retrieval method and represents the trend of the next-generation search engine. Existing QBH systems still have the problems that most of the retrieval algorithms are of slower searching speeds than the traditional query-by-text ones; lacks a practical QBH system based on the Masyas framework and the GPU acceleration algorithms. This paper introduces a query-by-humming system which is based on the Marsyas framework and GPU acceleration algorithms. The system design, the humming retrieval process, the humming signal feature extraction, the melody libraries establishment and corresponding retrieval algorithms, the GPU parallel speed-up algorithms, and the experiments are presented in detail. The experimental results verify the functionality and good performance of the proposed methods by the speedup of more than 10X with our GPU acceleration algorithms.

Keywords: Query by humming, Marsyas, search engine, GPU

1. Introduction

Content-Based Audio Information Retrieval (CBAIR) is the technology which uses the physical characteristics such as the audio amplitude, spectrum and etc., auditory characteristics such as pitch, tone, and semantic features such as word, melody to realize content-based audio information retrieval. Because of the advantage of more accurate expression of user's retrieval requirements than Query by Text (QBT), Query by Humming(QBH) is currently one of the most important music retrieval methods. The focus of this paper is the QBH system design and its GPU algorithms.

Based on our survey results of the existing QBH systems in Table 1, we can find out that after years of development, there are already a lot of commercial systems, research organizations and platforms for QBH. Many non-commercial scientific research platforms such as MIREX (Music Information Retrieval Evalution exchange) [1–4], music information retrieval frameworks include Marsyas [5], CLAM [6]have also been developed. However, Existing QBH systems (Table 1) still has the problems that most of the retrieval algorithms are of slower searching speeds than the traditional query-by-text, and the Marsyas open source framework, which only provides a library of QBH algorithms, lacks a runnable QBH system and GPU acceleration algorithms[7–15].

In view of the above existing problems, this paper proposes a QBH prototype system design and realization method based on Marsyas open source framework, and its GPU acceleration algorithms based on CUDA. More in detail, by the analysis of the audio-processing algorithm library of Marsyas, we first introduce a QBH System design method based on Marsyas, together with the searching process flow, the choices and integration of related modules in Marsyas in Section 2. The establishment of the melody library based on MIDI files, Melody matching algorithms based on EMD and DTW, and a DTW's GPU parallel speed-up algorithm are described in Section 3, Section 4 and Section 5. The experimental results verify the functionality and performance of the proposed methods which show that the speedup of more than 10 is achieved for our DTW GPU acceleration algorithms in Section 6. Section 7 concludes the paper.

* Corresponding author e-mail: ruanli@buaa.edu.cn

Table 1 Comparision of QBH systems

Name	Dev.Organizaion	Dev.Purpose	Scale	Technology	Open Source
Vocal-Search	UMich and CMU	Research	Hundred	String alignment Error probability model	No
M-MUSICS	Korea and Asia University	Commercial	Unknown	Mobile applications in C S mode	No
Midomi	Melodis Corporation	Commercial	Million	Unknown	No
Grand humming system V02	Grand innovation homes	Research	Hundred	Linear alignment	Yes
Humming soso	Tencent	Commercial	Million	Unknown	No

2. Overall Design of QBH System Based on Marsyas

The overall design of our QBH system based on Marsyas is shown in Fig.1. By carefully choosing, integrating and modifying modules in Marsyas, our system includes key modules like humming melody extraction, melody library establishment and rhythm matching, parallel acceleration to constitute a complete humming retrieval process from the hum sound input to search results feedback.

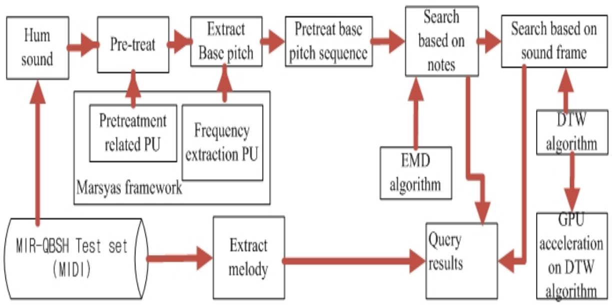


Figure 1 Overall system design

The features of the sound frame and notes are extracted from humming sound by choosing and integrating the corresponding algorithms in Marsyas. The melody library is established based on the MIR-QBSH test data set. After the feature extraction and the melody library establishment process, the query can be performed based on the melody library templates and humming melody templates. This paper presents a YIN (Hideki Kawahara et al.[4]) based signal feature extraction method and GPU based acceleration algorithms.

3. Humming Signal Feature Extraction

Humming signal feature extraction is the precondition for music retrieving and matching, and the extraction accuracy will directly affect the retrieval accuracy. Among various characteristics of the music, the pitch is the most important features of the music signal. The humming signal feature extraction include the following modules: pre-processing, pitch feature extraction, notes cutting, etc.

3.1. Humming Signal Preprocessing

The humming signal preprocessing process based on Marsyas is shown in Fig. 2. First, the humming input file which may be stored as a single channel or dual channel is read in and transformed to a single channel; Second, band-pass filtering is performed with an upper and lower cutoff frequency distance of between 60Hz and 5000Hz. Pre-emphasis processing is used to strengthen the high-frequency components; Next, the Hanning window is used for the audio frame signal; Finally, filtering according to the energy spectrum. The process is: calculate the energy of each frame $Mean(i)$ and the average energy of all the frames $Mean(All)$; If the i_{th} frame energy $Mean(i)$ is less than the average energy of Mean $(All)*0.3$, the i_{th} frame is defined as a silent frame and set to zero.

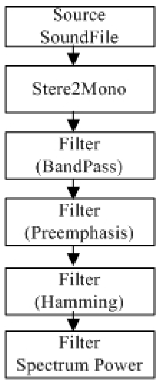


Figure 2 Humming signal preprocessing

3.2. An Improved YIN Algorithm based Pitch Feature Extraction

3.2.1. Pitch Feature Extraction Process

The proposed pitch feature extraction process based on the SCAF is shown in Fig. 3. In our process, the base frequency extraction algorithm of SACF, YIN and peak extraction algorithm of Peaker in Marsyas are chosen and integrated.

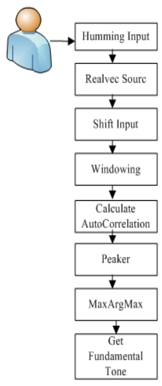


Figure 3 Pitch feature extraction process based on SACF

In Fig.3, *Realvec Source* is used to read the sample data matrix; *Shift Input* implements inter-frame overlap; *Windowing* adds Hanning window on frame to highlight the middle part characteristics in the frame; *AutoCorrelation* calculates the value of the autocorrelation function of the sampling signal; *Peaker* acts as a peak selector which selects the peak in the autocorrelation function; *MaxArgMax* returns the location of the peak. Through the series treatment, we get the pitch sequence of humming input signal. Based on Fig.3, our modified YIN algorithm based humming process is shown in Fig.4 where *Shift Input* achieves frame overlap and the YIN module outputs the audio frame pitch.

3.2.2. Humming Feature Extraction Algorithm based on YIN[4,7–12]

This section describes our humming feather extraction algorithm based on the YIN. We first define the humming signal x_t with periodic T which satisfies the following characteristics.

$$x_t - x_{t+T} = 0 \tag{1}$$

$$\sum_{j=t+1}^{t+W} (x_j - x_{j+t})^2 = 0 \tag{2}$$

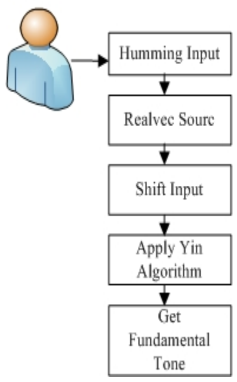


Figure 4 Pitch feature extraction process based on YIN algorithm

For an unknown period, by some modification we can get the difference function as Eq.3.

$$d_t(\tau) = \sum_{j=1}^{W} (x_j - x_{j+\tau})^2 \tag{3}$$

As a result, we get the cycle when $\tau = 0$.ACF is very sensitive to change of magnitude. The difference function solves this problem. The standardization of the difference function is shown in Eq.4

$$d_t^{'}(\tau) = \{^1 \frac{d_t(\tau)}{[(\frac{1}{\tau}\sum_{j=1}^{\tau} d_t(j))]} \tag{4}$$

By this means, it is possible to find the valley value near the zero displacement point which makes the extraction of high fundamental frequency values become possible. Therefore, the valley value can be discovered by setting an absolute threshold. Humming signal energy value in a frame can be calculated.

The valley points below the dotted line are used as the result set. Then local minimums are estimated by the parabolic difference method and the cycle is the interval between the valley values. The result of our YIN algorithm based extraction is shown in Fig.5.

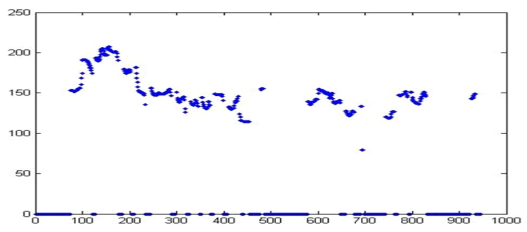


Figure 5 Feature extraction result

From Fig.5, we can find that our extraction method has few dirty points and can accurately extract the base frequency information which is beneficial to the next step- note cutting.

3.3. Notes Cutting

Notes cutting module is responsible for the transcription of the audio frame to note. The basic format of the resulted note is "*note values - note length*". The *note value* is of the MIDI format which refers to Chromatic Design with the difference between adjacent semitones value of 1 to facilitate comparison; the *note length* represents the number of audio frames. There are two kinds of cutting methods: cutting based on time domain waveforms and pitches.

3.3.1. Time Domain Waveform Based Cutting

Time-domain waveform energy represents note division information to some extent. For example, in Fig.6, there are obvious quiet zones in the waveform of the Chinese song of *March of the Volunteers* between each two words. According to these features, we can divide the notes in the time domain and then merge the same note. The most common method is to calculate the average energy and set its threshold as [0.1, 0.3]. The zone less than the threshold is regarded as the quiet zone. However, this approach has many drawbacks. For example, in Fig.7, due to the strong coherence of "de" and "di" in the singing, there is no quiet zone and we cannot get the correct information about the notes. Moreover, if the noise is strong, the quiet zone may also have a high energy; the method cannot be applied.

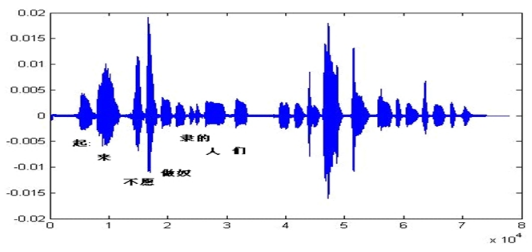


Figure 6 The time domain waveform of the *song of March of the Volunteers*

3.3.2. Pitch-based Cutting

As is shown in Fig.8, during humming, similar pitches render polymerization within a certain period. However,

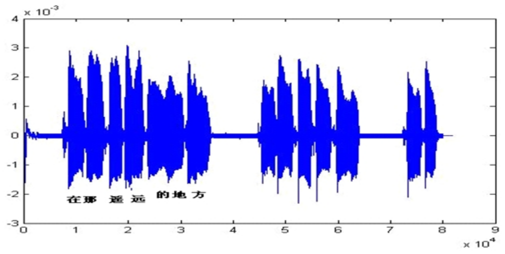


Figure 7 The domain waveforms in the song of *In That Distant Place*

although the humming user may use the same pitch in that period, because the vocal fold vibrations are not completely stable, and combined with the natural transition among notes, it is impossible to obtain a very uniform pitch contour. Therefore, further processing is required.

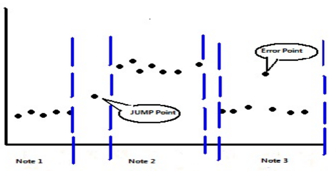


Figure 8 Original pitch value distribution

The flow of our pitch-based cutting method is shown Fig.9, which includes algorithms of pitch sequence input, dirty spots removal; mean filtering, median filter, mute fill, repeated spots removal. After that process, we get the uniform pitch sequence and the note values and note length information.

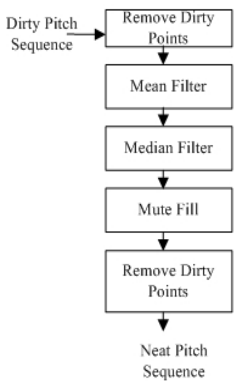


Figure 9 Transcription process of pitch to note

First, remove the mute point; then filter the dirty spots including the transition points and the error points. The detailed steps of dirty point removal algorithms by fitting adjacent points are: Set a threshold; if difference between the current point and the adjacent frames is greater than the threshold, the current point is regarded as a trip point or error point. Calculate the difference *DiffLeft* between the current point and the left point, and the difference *DiffRight* between the current point and the right point; Compare *DiffLeft* and *DiffRight* and set current point value to the smaller one. After this fitting, similar pitches will be normalized to the same pitch value. Merge every five pitch points and the value of a new pitch is to take the mean of these five points. Next, repeat the above fitting method to fit the new pitch sequence. So we get a normal pitch sequence, remove the notes whose length is less than 2, and fit the notes. The dirty spots removal process is shown in Fig.10.

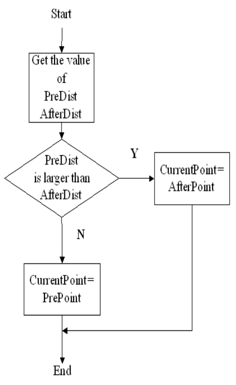

Start
Get the value of PreDist AfterDist
PreDist is larger than AfterDist
Y
N
CurrentPoint= AfterPoint
CurrentPoint= PrePoint
End

Figure 10 Dirty point removal flow

After our pitch-based cutting method, the resulted distribution for samples in Fig.8 is shown in Fig.11.

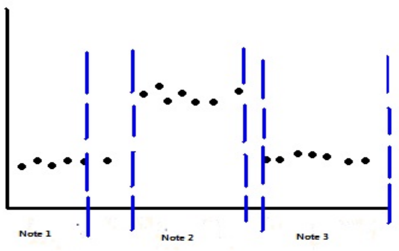


Figure 11 Pitch value distribution after fitting

The normalized note sequence is shown in Fig.12.

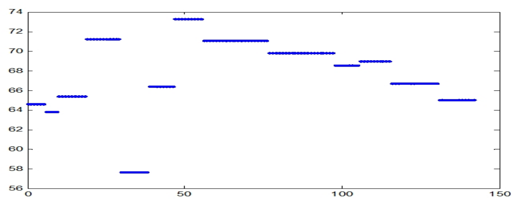


Figure 12 Normalized note sequence

4. Melody Library Establishment

4.1. Comparison of MIDI Files and Waveform Files

We use midi instead of wav, mp3, ogg wave files to build the library, because the multi-pitch melody extraction results were unsatisfactory with the highest accuracy rate of only about 80 percent, and the midi file direct recorded melody of the song which can be easily extracted from the melody of the song thus with a higher extraction rate.

4.2. Analysis of the MIDI Format

MIDI files are binary files and have the following basic structure: header and data description. Table 2 shows the structure of a MIDI file. *Type* refers to the *"MThd"* or *"MTrk"* where *MThd* represents records of the file header information and *MTrk* represents the record track information. The organization of MIDI files is: *header* + $track_1$ + $track_2$ +...+ $track_i$+...+ $track_n$.

Table 2 Block structure of the MIDI file

Type	Length	Data
4-bytes	4-bytes	4-bytes

4.3. Melody Library Establishment Process

The melody library establishment process includes the following steps:

Step1 (Extract and preprocess the track information): We extract the track information which contains notes, the start event and end event of notes from the MIDI file. To ensure the accuracy of the music library, we set the main theme in track 0 and delete the other tracks.

Step2 (Divide the File Into Phases): To satisfy people's custom of recording the song by the unit of phrase, reduce the granularity of the search, improve the retrieval speed, and also ensure that user can search when he/she sings from the middle part of the music, we divide the file by phases. Each phrase contains the sequence and the duration of the notes.

Step3 (Establish the Melody Records of the Library): Integrates the phases to establish a number of records of phrases in the melody library. An example is shown in Fig.13 where the solid blocks represents notes, the vertical axis indicates the note pitch, the horizontal axis represents timeline, and the location of the note indicates that the time period when the note is sound. The value and length of the pitch of the note sequence constitutes a melody contour.

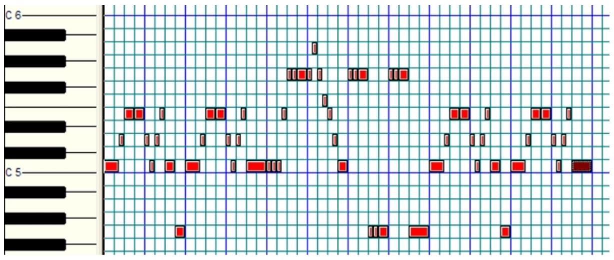
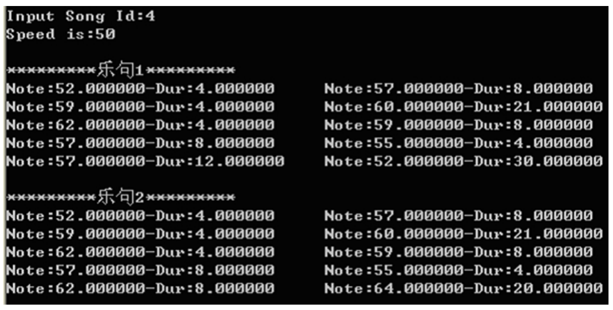


Figure 13 Note information recorded in MIDI

Fig.14 presents an example real-time playback information based on our phase based melody library under the Windows platform using the winmm.dll system function $midi\_outshort()$.



```
Input Song Id:4
Speed is:50

**********乐句1**********
Note:52.000000-Dur:4.000000      Note:57.000000-Dur:8.000000
Note:59.000000-Dur:4.000000      Note:60.000000-Dur:21.000000
Note:62.000000-Dur:4.000000      Note:59.000000-Dur:8.000000
Note:57.000000-Dur:8.000000      Note:55.000000-Dur:4.000000
Note:57.000000-Dur:12.000000     Note:52.000000-Dur:30.000000

**********乐句2**********
Note:52.000000-Dur:4.000000      Note:57.000000-Dur:8.000000
Note:59.000000-Dur:4.000000      Note:60.000000-Dur:21.000000
Note:62.000000-Dur:4.000000      Note:59.000000-Dur:8.000000
Note:57.000000-Dur:8.000000      Note:55.000000-Dur:4.000000
Note:62.000000-Dur:8.000000      Note:64.000000-Dur:20.000000
```

Figure 14 Example of real-time play of phase based melody library

5. Melody Matching and GPU Acceleration Algorithms

5.1. Humming Melody Retrieval Algorithm based on EMD (HEMD)

5.1.1. Design of HEMD

This paper introduces a humming melody retrieval algorithm based on EMD (Earth Moving Distance) called HEMD. The basic idea is that : The length of the node is denoted as the transportation unit; establish transportation cost table taking both the value and length of the notes into consideration; count out the optimal transportation strategy and minimum cost. The music which requires the least transportation cost is the best mapping object. The humming template is modeled as *Earth* and should be move to a given position. Fig.15 is an example where *BB'* is a note of the humming template which is the source. The vertical axis is the pitch value and the horizontal axis is time. The length of *BB'* on the vertical axis represents the length of the note and the transportation unit in the original EMD model. *AA'* represents a note in the humming template which is the destination.

The transportation cost table should be configured to reflect the correspondence between the pitch value and the position of the note. That is, to guarantee the purpose of *BB* ' in the transportation program is *AA'*. Euclidean distance can reach the above results.

In this way, the original shipping table is change into a cost table from the humming template to the song template; algorithm finds the minimum cost and its value.

5.1.2. Complexity analysis

The limitation of HEMD algorithm is its high space complexity, and needs *M*N* space to record the cost of the table, which *M* is the number of notes in humming template, *N* is the number of notes in music template. Due to the limitations of program space, the length of humming template and music template cannot be too long, to the 10s humming sound audio and frame information can reach the 1000 notes after the segmentation of only a few dozen, and this is to select the note information as reason of the EMD algorithm parameters.

Audio frame information of the humming sound in 10s can reach 1000, the notes after the segmentation are only a few dozen, and this is why select the note information as the EMD algorithm parameters.

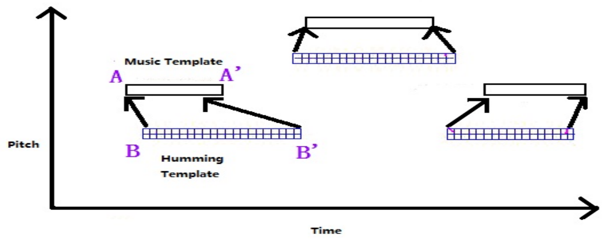


Figure 15 Model of HEMD

5.2. Humming Melody Retrieval Algorithm based on DTW (HDTW)

5.2.1. Design of HDTW

Original DTW algorithm is an approximate string matching algorithm based on dynamic programming. The essence of DTW is to get the shortest distance between the strings. In our QBH context and the following description of HDTW, the *Destination* and *Source* corresponds to the feature vectors in music templates and humming templates. Each feature vector is made up of the one-dimensional features of the pitch values.

The shortest cumulative distance of HDTW expressed the degree of similarity between humming template and music template. The smaller the cumulative distance is, the more similar the template is.

Definition 1: The known music template (*destination* string) of length m is denotes as *Destination*=$\{D(1), D(2), \ldots, D(i), \ldots, D(m)\}$.

Definition 2: The humming template (*source string*) of length n is denotes as *Source*=$\{S(1), S(2), \ldots, S(j), \ldots, S(n)\}$.

Definition 3: The index which points to *Destination* is denoted as *IndexD*.

Definition 4: The index sequence which points to *Destination* is denoted as *IndexDS*=$\{$ *IndexDS(1)*,…, *IndexDS(k)*,…, *IndexDS(Id)* $\}$.

Definition 5: The index which points to *Source* is denoted as *IndexS*.

Definition 6: The index sequence which points to *Source* is denoted as *IndexSS*=$\{$ *IndexSS(1)*,…, *IndexSS(k)*,…, *IndexSS(Is)* $\}$.

Therefore, based on the above definitions, the HDTW algorithm is introduced in Algorithm1 and the flowchart in Fig.16.

```
Algorithm1: HDTW
Input: Destination, Source, IndexD, IndexS.
Output: IndexSS and IndexDS.
1.Initialize IndexD(1) =0;IndexS(1)=0; // Initialize IndexD(1) and IndexS(1) to the first character (element) of Destination and Source.
2. While not the last character of the Destination and Source do {
3. If S(IndexS) = D(IndexD) then
    4.{If S(IndexS +1) = D(IndexD) then IndexS++;
    IndexSS= IndexS ⋃ IndexS ;
    goto step 3;}
    5. else {If S(IndexS) = D(IndexD+1) then IndexD++;
    IndexDS=IndexDS ⋃ IndexD ;
    goto step 3; }}}
    6. Return IndexSS and Index DS.
```

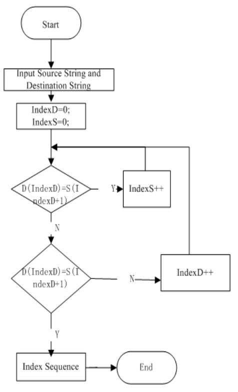


Figure 16 Flow of HDTW algorithm

As HDTW algorithm is based on dynamic programming; therefore, the sequences of the Index values IndexSS and IndexDS will form a shortest path from source string to destination string which can be expressed by the two-dimensional plane (Fig.17).

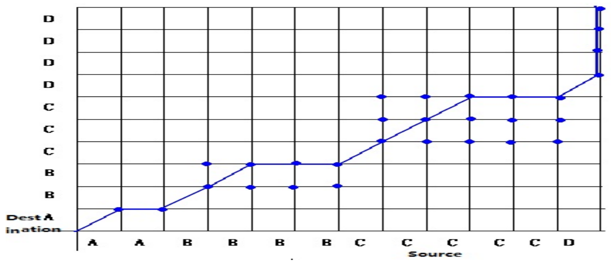


Figure 17 2D plane shortest paths

In Fig.17, the vertical axis is *Destination* and the horizontal axis is *Source*. The matching process path will be a two-dimensional vector (*Source*, *Destination*) sequence. Each point in the grid represents the equality between the character in Source and that in Destination. The polylines represent the shortest path through equivalent points to Destination. Fig.18 is an example by completely mapping into QBH.

Definition 7: A two-dimensional vector sequence $\phi(S,D)$ is used to denote the point in the shortest path.

Definition 8: The shortest path ShortPath $=\{\phi(S_1,D_1)\phi(S_2,D_2),\ldots,\phi(S_n,D_n)\}$.

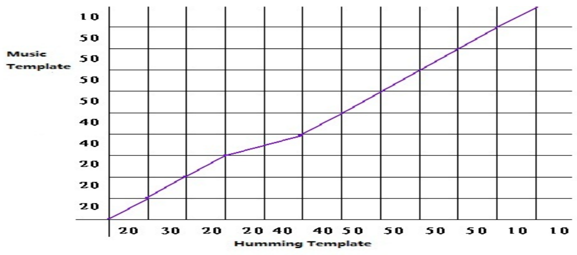


Figure 18 The meaning of the shortest path in the two-dimensional plane

The slope is restrict as [0.5 , 2] to maintain that path is always toward the upper right. So after $\phi(S_i,D_j)$, the next grid can only be one point among$\{\phi(S_{i+1},D_{j+1})$, $\phi(S_{i+1},D_{j+2})$, $\phi(S_{i+1},D_{j+2})\}$.

So HDTW is through the method of dynamic programming to calculate minimum cumulative distance which the path goes through each grid point. The minimum cumulative distance that from lower left to upper right can be always found by the Eq.5 and Eq.6.

$$CumDist(\phi(s_i,D_j)) = Dist(\phi(s_i,D_j)) + PreMinDist \quad (5)$$

$$PreMinDist = Min(CumDist(t\phi(s_{i-1},D_{j-2})),$$
$$CumDist(t\phi(s_{i-1},D_{j-1})),CumDist(t\phi(s_{i-2},D_{j-1}))) \quad (6)$$

To calculate the current cumulative distance *PreMinDist*, only three grid points around to accumulate distance value is needed. The value of the upper-right corner is the final cumulative distance value.

The basic steps to computing the similarity between humming template and music template are as following:

Step 1: An original distance matrix (Primary Matrix) is used to record the Euclidean distance Dist between pitch value$HumValue(i)$ at the humming template position i, and the pitch value $SongValue(j)$ at the song template position j using Eq.7.

$$Dist = |HumValue(i) - SongValue(j)| \quad (7)$$

Step 2: From lower left to upper right, followed by calculation of the cumulative distance of each grid point in each row, we finally get the cumulative distance matrix (Accumulate Matrix).

Step3: Select minimum value in the top row of the matrix from the cumulative value as the final HDTW shortest path distance.

To sum up, by solving the problem of different lengths of humming templates, we solve the mismatch problem caused by the stretching of the template in the timeline direction.

5.2.2. Complexity Analysis

Cumulative distance matrix is $M * N$ dimension where M is the length of the humming template, N is the length of the song template. Therefore, its space complexity is also very high. However, because the points in the cumulative distance matrix calculation only need the nearest points to participate in computing, and especially under the slope restriction of 0.5 to 2, only the cumulative matrix of the current row and the previous two lines, the computing has been localized. As is shown in Fig.19, the computation of D3 distance only depends on the minimum value of *D0*, *D1*, *D2* which are stored in the two lines before *D3*. By this means, the space complexity of HDTW is deduced.

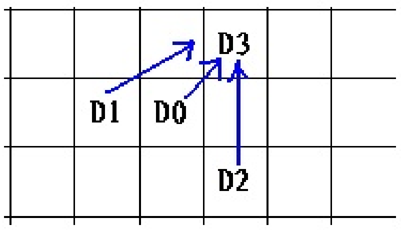


Figure 19 Calculation locality of HDTW

5.3. GPU acceleration algorithm for HDTW

Due to the calculation locality of HDTW, its space complexity is not so high which makes it possible to perform parallel acceleration. Common parallel acceleration of compute-intensive applications is through the GPU. A very good floating-point computing power of GPU will have a good accelerated effect for compute-intensive algorithm. Therefore, we design a GPU- accelerated algorithm for the HDTW based humming retrieval process.

Because there are only one humming template and multiple song templates to be matched, and there is no dependence among the DTW computations for different songs, we implement our GPU acceleration on music. The basic process is: we use Nvdia's CUDA framework, design different kernel functions for key modules in HDTW and perform GPU acceleration for the key parts of the algorithm. As an example, the pseudocode of our kernel function for the part of the accumulation matrix calculation in the HDTW algorithm is as shown in Algorithm 2.

Algorithm 2: HDTW parallel algorithm
Input: Humming melody template *H*;
The melody Library templates *L(1)L(2)...L(n)*;
Output:DTW distance results *RES[1,...,n]*

```
For $I$ =1 to n in Parallel
    $RES[i]=CalDtw( H , L(n) )$
Endfor
EndAlgorithm
```

The melody matching process between the different templates is parallel computed. After the parallel computation, the results are uniformly processed in the main function(Fig.20).

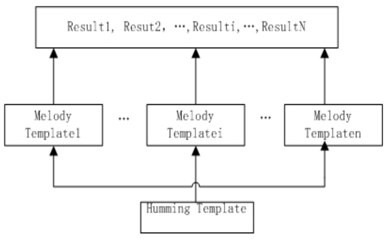


Figure 20 Parallel Processing in humming retrieval of HDTW algorithm

6. Experiments and Results Analysis

6.1. Test Data Set and Evaluation Metrics

The standard QBH testing set MIREX2003 is chosen so as to evaluate the system in metrics of searching accuracy and efficiency.

The test process and effect are that: (1)The user hums a piece of music; (2) the system returns a list which shows the first N results from the rank according to the extent of similarity to user's piece.

In general, there are two metrics to evaluate the accuracy of the humming searching system: Mean Reciprocal Rank and hitting ratio of the first *N* results.

We also choose the commonly used metrics MRR (Mean Reciprocal Rank) , which is also taken by MIREX as the performance evaluation metric, in our experiments. MRR is defined in Eq.8s where the value of n represents the number of pieces of music, and the value *i* represents the rank of results.

$$MRR = \sum_{i=1}^{n} \frac{1}{Rank(i)} \tag{8}$$

Hitting ratio of the first *N* is defined in Eq.9.

$$Top(N) = \frac{Hit(N)}{Total} \tag{9}$$

Hit(N) represents the times of potential target ranking first N results, and Total is the number of tests. Accuracy of our system is tested by hitting ratio of the first *N* results.

Apart from the accuracy of the system, run time of a system is also an important performance evaluation metric. When the humming searching system contains millions of songs, run time could be extremely essential due to the scale of data.

6.2. Results Analysis

6.2.1. Evaluation of system performance

We use the humming searching system public testing dataset published by National TsingHua University. The testing set contains 48 pieces of MIDI songs with humming results by different people in different years. person00001, person00002, person00003, person00004 and person00006 in year 2003 are selected to run our HDTW and HEMD in our test. Results are shown in Table 3.

As is shown on the table, the hitting ration of HDTW algorithm is higher than the HEMD algorithm. Besides, the Mean Ranks of the top ten songs by HDTW algorithm is lower than that of HEMD. This is due to that HEMD is based on note while HDTW on frame. Therefore HDTW achieves a better functional performance than HEMD.

MIREX-2003-person00001-00013.mid has a good searching performance and achieves the highest score(Fig.21,Fig.22). Moreover, the revised humming model and music model note perfectly match each other. However, for MIREX-2003-person00002-00029.mid, the revised wave shape is shown as Fig.23 . When the time is equal to 90, there is a huge difference between the real note and the corresponding music model note, which leads it only ranks at 10. In other words, a big disparity between notes can contribute big influence in results due to a larger calculation scale in HEMD.

From Table 3, we also find that the searching results vary with different uses. For example, the lowest hitting ration in top 10 results is 15% while the highest one is 89%. This hitting ratio is related to the accuracy of the humming piece. Some users can control the melody more accurate, while others are poor at this skill.

Moreover, we also find that the searching results depend on the characteristic of the music. For example, the music 00039 (see Fig.24) has a high frequent and dramatic variance in music notes, and therefore, error notes can easily happen although the basic melody can be hummed out. To solve this problem, removing the outlier notes should be taken into consideration.

6.3. Running Efficiency Analysis

The testing results of the run time of the two algorithms are shown as Fig.25. We can find that the run time of HDTW is ten times more than HEMD.

Table 3 Testing results of MIREX 2003

Test person	Algorithms	TOP3 Hit rate	TOP5 Hit rate	TOP10 Hit rate	TOP10 Average rank
MIREX 2003 person00001	EMD	0.64	0.76	0.81	2.18
	DTW	0.81	0.90	0.95	1.30
MIREX 2003 person00002	EMD	0.33	0.43	0.52	3.91
	DTW	0.38	0.52	0.76	2.14
MIREX 2003 person00003	EMD	0.42	0.58	0.68	4.00
	DTW	0.68	0.79	0.84	1.50
MIREX 2003 person00004	EMD	0.05	0.15	0.20	5.5
	DTW	0.40	0.55	0.65	2.84
MIREX 2003 person00006	EMD	0.40	0.52	0.60	3.38
	DTW	0.81	0.85	0.89	1.71

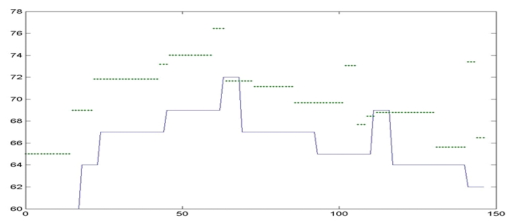


Figure 21 MIREX-2003-Person00001-00013.mid Original humming model and musical wave model

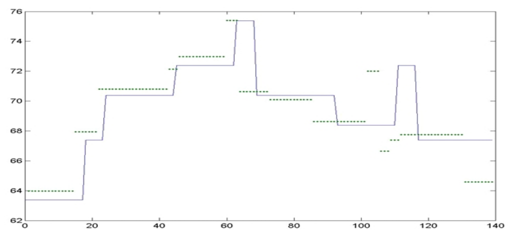


Figure 22 MIREX-2003-Person00001-00013.mid revised humming model and musical wave model

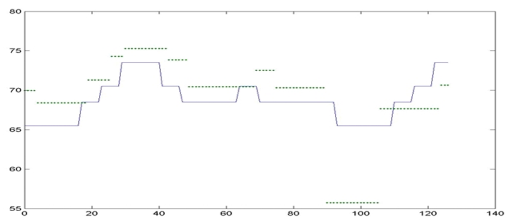


Figure 23 MIREX-2003-Person00002-00029.mid Revised humming model and musical wave model

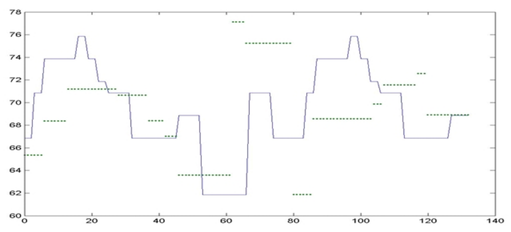


Figure 24 MIREX-2003-Person00002-00039.mid Revised humming model and musical wave model

6.3.1. GPU Acceleration Performance Analysis on HDTW

To test the GPU acceleration performance on HDTW, the acceleration ratios are computed and the results are shown in Fig.26. We can find out that the larger of the scale of the music library, the higher of the accelerating ratio. When the scale is 5000 songs, the accelerating ratio is up to 10. In this way, the low speed of HDTW is compensated. When the scale of the music library is larger than 5000 songs, GPU-accelerated HDTW is of better performance than GPU-accelerated HEMD. Therefore, GPU-accelerated HDTW is recommended as a more reasonable choice when the music library is of large-scale.

7. Conclusion

Query-by-humming(QBH) represents the trend of the next-generation search engine. Existing QBH system still has the problems of slow searching speeds and lacking a practical QBH system based on Marsyas. This paper introduces a QBH system which is based on the Marsyas framework and GPU acceleration algorithms. The experiments show that our method is of great performance and the GPU acceleration method can save

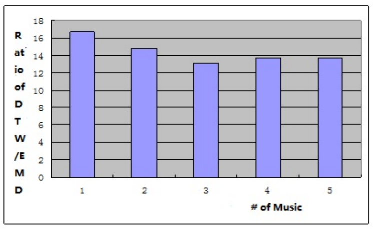


Figure 25 HDTW/HEMD Ration of Searching time

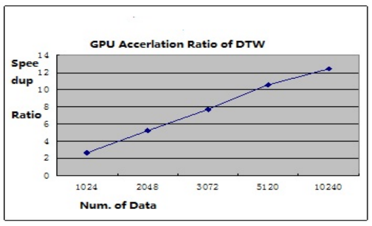


Figure 26 Acceleration ratios of HDTW

more than 10 times of searching time. We are now researching on the performance improvement methods on a wave-file based QBH system and its GPU acceleration methods.

Acknowledgement

This work was supported by the Hi-tech Research and Development Program of China (863 Program) under Grant No. 2011AA01A205, National Natural Science Foundation of China under Grant No. 61003015, 60973008 and 60973007, Beijing Natural Science Foundation under Grant No. 4122042, the fund of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2012ZX-23 and the Fund of Beijng Nova Program under grant NO. Z121101002512087.We express our gratitude for the related students (especially, Yao Zhen, Hairong Mu) in Beihang University who participated in our project.
The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] Ahyoung Lee and Ilkyeun Ra.A Queuing Network Model Based on Ad Hoc Routing Networks for Multimedia Communications, Appl. Math. Inf. Sci. Vol. **6** No. 1S (2012) pp. 271S-284S
[2] Casey, M.A.,Veltkamp, R., Goto, M.; Leman, M.etc.. Content-Based Music Information Retrieval: Current Directions and Future Challenges. Proceedings of the IEEE . Vol. **96**, no. 4 (2008). pp. 668-696
[3] K. Kim, K. R. Park, S.J.Park,etc..Robust Query-by-singing/humming System Against Background Noise Environments. IEEE Transactions on Consumer Electronics, vol. **57**, no. 2, pp. 720-725, May 2011.
[4] A de Cheveign, H Kawahara. YIN: A Fundamental Frequency Estimator for Speech and Music. The Journal of the Acoustical Society of America. Vol. **111** No. 4 (2002). pp. 1917-1930.
[5] http://marsyas.info/.
[6] http://clam-project.org/.
[7] A. Ghias, J. Logan, etc.. Query by Humming: Musical Information Retrieval in an Audio Database. In Proceedings of the third ACM international conference on Multimedia (MULTIMEDIA ’95). ACM, New York , USA. (1995) 231-236.
[8] N.H. Adams, M.A. Bartsch, G.H. Wakefield,.Note Segmentation and Quantization for Music Information Retrieval IEEE Transactions on Audio, Speech, and Language Processing, vol. **14**, no. 1, pp. 131-141, Jan. 2006.
[9] Anssi P. Klapuri , Antti J. Eronen and Jaakko T. Astola. Analysis of the Meter of Acoustic Musical Signals. IEEE Trans. Speech and Audio Processing (2004). pp. 342-355.
[10] X.H. Chen, MIDI principles and the development and application, National Defence Industry Press, (2008).
[11] X.Q. Yu, G.W Wan. New method of speech recognition based on auditory spectral characteristics. Chinese Science Abstracts, **4(3)**: 374-375(1998).
[12] W. Golab and R. Boutaba, Policy-driven automated reconfiguration for performance management in WDM optical networks, IEEE Commun. Mag. **42**, 44-51 (2004).
[13] L. Wang. Design and implement of a Singing or Humming based music retrieval system. [Bachelor Thesis] Supervised by Li Ruan. Beihang University. 2011.
[14] L. Ruan, M.F. Zhu, L.M. Xiao. Content Addressable Storage Optimization for Desktop Virtualization based Disaster Backup Storage SystemChina Communication. **19(7)**: 1-13(2012).
[15] A. Balinsky and N. Mohammad. Non-Linear Filter Response Distributions of Natural Images and Applications in Image Processing, Appl. Math. Inf. Sci. **5**, 3 (2011) pp. 367-389

Dr. Li Ruan is currently a senior member of the China Computer Federation and an assistant professor in the institute of computer architecture, school of computer science and engineering at Beihang University, Beijing, China from 2009. Her research interests are system virtualization, high performance storage and big data. She received her Master and Ph.D. from Chinese Academy of Science in 2004 and 2009.

Long Wang received his B.S. in school of computer science and engineering in Beihang University, Beijing, China in 2012. His research interests are computer system software and parallel computing.

Dr. LiMin Xiao is currently a senior member of the China Computer Federation and a professor in the institute of computer architecture,School of Computer Science and Engineering, Beihang University. His main research areas are computer architecture, computer system software, high performance computing. He received his BS degree in Tsinghua University, and his Master and Ph.D. from Institute of Computing, Chinese Academy of Science in 1996 and 1998.

Dr. MingFa Zhu is currently a professor in the School of Computer Science and Engineering at Beihang University. His current research areas are computer architecture, parallel computing, high performance computer system and network, and artificial intelligence. He received his BS degree in the School of Physics, Peking University in 1970, and his Ph.D. from Wayne State University in 1985.

Dr. YiLei Wu received the B.S. degree in electronic engineering from Fudan university, Shanghai, China, in 1998, the Ph.D. degree in automatic control from the Nanyang Technological University, in 2008. From 2005 to 2010, he was an senior engineer at Seagate Technology International, Singapore, before joining Beijing Computing Center as an engineering manager, in 2010. He is the first inventor of a few cloud system patents and has authorized many referred international journal publications. He is also an active industrial consultant and principal investigator of research programs targeted to industry in the area of big data.