VOICE QUERY TRANSCRIPTION AND EXPANSION SCHEME FOR EFFICIENT MUSIC RETRIEVAL*

Seungmin Rho[1], Byeong-jun, Han[2], Eenjun Hwang[2], Minkoo Kim[1]

[1] Graduate School of Information and Communication, Ajou University, Suwon, Korea
{anycall, minkoo}@ajou.ac.kr
[2] School of Electrical Engineering, Korea University, Seoul, Korea
{hbj1147, ehwang04}@korea.ac.kr

ABSTRACT

In this paper, we present a scheme for efficient humming-based music retrieval. For that purpose, we first describe how to extract a sequence of pitch and duration pairs as its feature information from sung or hummed query accurately and automatically. And then, we propose a novel scheme for reformulating user query to improve retrieval performance. The scheme is based on user relevance feedback with genetic algorithm. We implemented a prototype system based on these scheme and performed various experiments. Experimental result shows that our proposed scheme achieves an impressive performance.

1. INTRODUCTION

Most traditional approaches for retrieving multimedia data were based on the text information. However, in the case of content-based music retrieval, an easy and intuitive way to query a music library would be whistling, singing or humming a short fragment of a song. As melody is one of the most prominent and distinctive features of music, this short fragment could be a good clue to retrieving similar songs from a music database. For the efficient retrieval, melodies are analyzed and represented in the textual description such as UDR/LSR string or MIDI. So far, most research on content-based music retrieval has used symbolic representation with input ranging from note sequences using CMN (Common Music Notation) to user-hummed tunes [1].

One of the most popular interfaces for musical content is the QBH (Query-by-Humming), which enables even non-professional users to query just with some memorable tunes. The quality of QBH is strictly dependent on the accuracy of the audio translation such as pitch or duration of each note. Thus, an efficient algorithm to translate the audio signal into note-like representation is one of the crucial elements in QBH.

There are many techniques to analyze and extract pitch contour, interval and duration from vocal queries. In general, methods for detecting pitch and duration of music can be divided roughly into two categories; the time-domain based and the frequency-domain based. In the time-domain based methods, techniques such as ZCR (Zero Crossing Rate) and autocorrelation are popular [2]. The most frequently used method in the frequency-domain is FFT (Fast Fourier Transform) that is based on the property that every waveform can be divided into simple sine waves.

Query reformulation has been suggested as an effective way to improve retrieval efficiency in the text IR area and one of the well-known techniques for query reformulation is user relevance feedback. Recently, there has been an increased interest in the query reformulation using relevance feedback with evolutionary techniques such as genetic algorithm for multimedia information retrieval [3]. However, these techniques have still not been adopted widely in the field of music retrieval.

In this paper, we first propose a novel scheme that can transcribe user hummed query into notes more accurately than that proposed in our previous work [4]. The scheme is mainly based on methods called WAE (Windowed Average Energy) and ADF (Amplitude-based Difference Function) which is an improved version of AF (Amplitude Function). We also present a novel music retrieval scheme which supports user relevance feedback mechanism based on GA (Genetic Algorithm) to improve the quality of query results by reformulating user query.

The remainder of this paper is organized as follows. In Section 2, we present how to extract feature information such as pitch and duration from voice query automatically. In Section 3, we describe the query reformulation scheme and the prototype system we have built based on these two schemes. In Section 4, we report some of the experimental results. In Section 5, we conclude this paper.

2. VOCAL QUERY TRANSCRIPTION

* This research was supported by the MIC, Korea, under the ITRC support program supervised by the IITA. (IITA-2006-(C1090-0603-0002))and this research is also supported by the Ubiquitous Computing and Network (UCN) Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

This section describes "AMTranscriber," one of the key modules in the system which automatically extracts feature information such as pitch and duration from vocal queries. Based on the feature information, it generates symbolic notation such as UDR string for further processing.

2.1. Overall architecture

Figure 1 shows the overall architecture of the AMTranscriber for processing voice query. To handle a voice query, it first preprocesses the query using WAE and ADF. After that, notes are analyzed and their pitch and duration features are extracted.

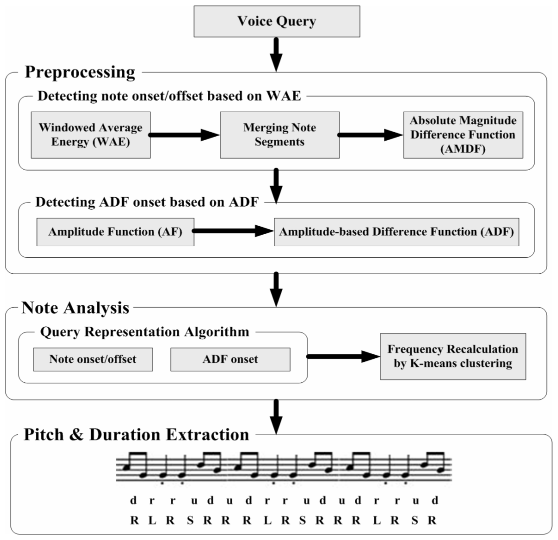

Voice Query
Preprocessing
Detecting note onset/offset based on WAE
Windowed Average Energy (WAE)
Merging Note Segments
Absolute Magnitude Difference Function (AMDF)
Detecting ADF onset based on ADF
Amplitude Function (AF)
Amplitude-based Difference Function (ADF)
Note Analysis
Query Representation Algorithm
Note onset/offset
ADF onset
Frequency Recalculation by K-means clustering
Pitch & Duration Extraction
d r r u d u d r r u d u d r r u d
R L R S R R R L R S R R R L R S R

Figure 1. Processing flow of AMTranscriber

To classify silent and voiced frames after framing, we use WAE and a heuristic method for merging discursively detected segments. Also, we use the AMDF (Average Magnitude Difference Function) to get fundamental frequency of each frame. Using this information, we finally get note onset/offset information.

2.2. Query processing

AMTranscriber uses two types of onset and offset information. The first type is note onset/offset which can be defined as follows: The moment when user starts to sing is defined as note onset; the moment when user stops singing is defined as note offset. The second type is ADF onset. ADF onset occurs when energy goes up rapidly from a low level or long silence.

In order to detect the note onset and offset, we applied new feature called WAE. The WAE is an improved version of the AE (Average Energy) which is a traditional energy estimation method. The AE itself indicates the average amount of energy of some signal range. This can be used to classify silent and voiced frames using one global threshold. In the WAE, a local threshold is defined for each window to classify silent/voiced frames. This could improve the accuracy of frame discrimination. Figure 2 shows the procedure for calculating local thresholds for frames according to the WAE.

The output from the WAE contains many continuous and tiny frame segments. In order to restore the continuity between frame segments, we may need to merge some of them. In this paper, we defined a segment with a minimal length for further analysis as meaningful. With this definition, we merged tiny segments by converting a vocal segment of short length to a silent segment and merging neighboring silent/vocal segments. After this, AMDF was applied to compute the fundamental frequency of each classified frame for further usage.

```
Procedure LocalThbyWAE(AE, WinSize, GlobalTh, r)
Step 1: For each AE(i) in unit window whose size
           is WinSize,
              repeat Step 2 ~ Step 3
Step 2: NewTh ← r × Maximum of AE in unit window
Step 3: If AE(i) > GlobalTh and AE(i) > NewTh
           Then LocalTh(i) ← NewTh
           Else, LocalTh(i) ← GlobalTh
```

Figure 2. Main procedure of WAE

In order to detect an ADF onset, AF is computed in the first phase. After this, AD and ADF are analyzed. After detecting the note onset/offset and ADF onset, we can obtain more accurate note segment information. The algorithm for computing ADF is shown in Figure 3. Continuous positive and negative differences are summed up for the whole signal. That is, continuously increasing/decreasing intervals are merged into an increasing/decreasing interval.

```
1. For each frame, compute the Amplitude
   Function(AF)   $AF(k) = \sum_{i=1}^{FrameSize} |x(k)|$
2. Compute the Amplitude-based Difference
   Function, ADF(k)=AF(k+1)-AF(k)(1≤n≤#OfFrames-1)
3. Find continuous positive ADF
   from n_start to n_end
4. Compute the accumulated value AV
   $AV = \sum_{i=n\_start}^{n\_end} ADF(i)$
5. Set ADF(n_start)=AV, others set to 0
6. If ADF(n) is negative, set ADF(n)=0
```

Figure 3. ADF computation algorithm

Finally, it is essential to recalculate the frequency of note, since the fundamental frequencies of each frame are different within a note segment. Here, we applied the K-Means clustering method for clustering the result of AMDF.

3. QUERY REFORMULATION AND IMPLEMENTATION

In this section we will describe a GA-based scheme for user query refinement, query interfaces and overall architecture of our prototype system.

3.1. Query refinement

As we mentioned earlier, in this paper, we implemented a GA-based relevance feedback scheme to improve retrieval performance.

Our GA starts with initializing a population and then evaluates this population using a fitness function that returns a value for each chromosome indicating its quality as a solution to the problem.

We calculate the fitness value of the chromosome using the following formula:

$$Fitness = \frac{1}{\sum_{i=1}^{N} relevance(M_i)} \sum_{i=1}^{N} \frac{1}{i} \sum_{j=1}^{i} relevance(M_j) \qquad (1)$$

Here, N is the total number of music objects retrieved in population P and relevance(M) is a function that returns the relevance of the music M. The equation for the relevance function is:

$$relevance(M) = \frac{\sum_{i=1}^{n} \frac{QueryLength - Cost(LD)}{QueryLength}}{n} \qquad (2)$$

Here, M is the music object in population P, n is the number of matched melody segments in each music object M. The relevance value ranges from 0 to 1, where '1' indicates that the music is relevant to the user query with full confidence and '0' indicates the opposite case. The Levenshtein Distance (LD) [5] is a measurement of the distance between two strings by the number of deletions, insertions, or substitutions needed to make them equal.

The GA produces a new generation iteratively before it terminates. At each generation, chromosomes are selected by a tournament selection method with tournament size of 5. We randomly choose 5 chromosomes with equal probability from the population. We use the classical single-point crossover and mutation which is implemented as a random process. A random number is generated in a given interval, in our case [-10, 10], and that number is taken as the new value for the gene that has to mutate.

3.2. System architecture

We have implemented a prototype music retrieval system based on the features that we described earlier.

Figure 4 shows the overall system architecture of our prototype system for processing user query. The system consists of three main components: Interface, Analyzer, and GA Engine. Typical query processing scenario is as follows: User first makes an initial query using one of the four different user query interfaces: QBC (Query by Contour), QBH (Query by Humming), QBE (Query by Example) and QBMN (Query by Music Notation). When a user query is given, Analyzer module interprets the query as a signal or a sequence of notes, and extracts audio features such as pitch and time contour. Then, those extracted features are transcribed into uUdDr and LSR string. For the transcribed string, the FAI index, which collects frequently queried melody tunes intelligently for fast query matching, is first looked up and then the music database is searched for when the index lookup failed. A detailed description for the FAI (Frequently Accessed Index) indexing scheme can be found in Rho and Hwang [1].

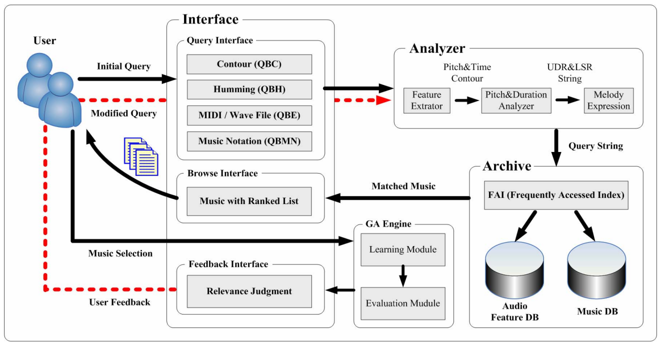
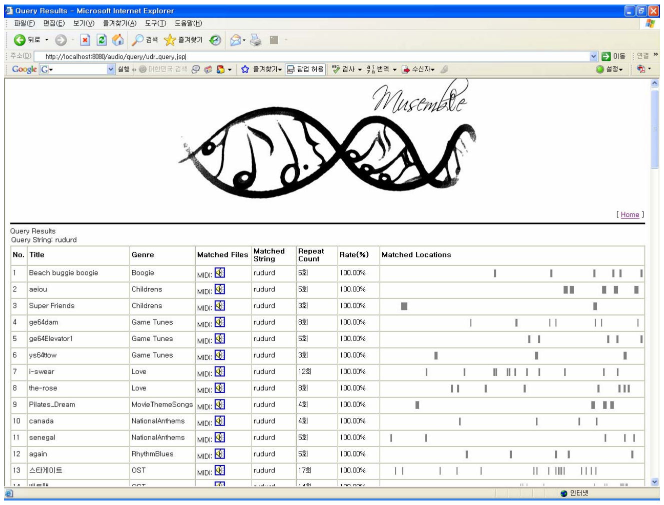


Figure 4. System architecture of MUSEMBLE

The matched melodies are displayed according to their rank on the browse interface. When the user selects a melody or its segment as the most relevant one, GA engine generates new music segments and evaluates fitness of each segment based on our genetic algorithm. A modified query is generated by the user's relevance judgment via feedback interface, and then the whole query process will be repeated until the user is satisfied.

3.3. Interfaces

Figure 5 shows a screen snapshot where matched music from the FAI index and music database for queried melody are displayed. Results in the matched list are ranked and listed in the descending order of their similarity to the query. As shown in Figure 5, user can easily playback the matched melody segments of each music object by clicking the grey bar on the matched location area.


Query Results - Microsoft Internet Explorer
http://localhost:8080/audio/query/udr_query.jsp
Musemble
[Home]

Query Results
Query String: rudurd

No.	Title	Genre	Matched Files	Matched String	Repeat Count	Rate(%)	Matched Locations
1	Beach buggie boogie	Boogie	MIDI:	rudurd	6회	100.00%	
2	aeiou	Childrens	MIDI:	rudurd	5회	100.00%	
3	Super Friends	Childrens	MIDI:	rudurd	3회	100.00%	
4	ge64dam	Game Tunes	MIDI:	rudurd	8회	100.00%	
5	ge64Elevator1	Game Tunes	MIDI:	rudurd	5회	100.00%	
6	ys64ttow	Game Tunes	MIDI:	rudurd	3회	100.00%	
7	i-swear	Love	MIDI:	rudurd	12회	100.00%	
8	the-rose	Love	MIDI:	rudurd	8회	100.00%	
9	Pilates_Dream	MovieThemeSongs	MIDI:	rudurd	4회	100.00%	
10	canada	NationalAnthems	MIDI:	rudurd	4회	100.00%	
11	senegal	NationalAnthems	MIDI:	rudurd	5회	100.00%	
12	again	RhythmBlues	MIDI:	rudurd	5회	100.00%	
13	스타게이트	OST	MIDI:	rudurd	17회	100.00%	

Figure 5. Query result interface

The original query will be displayed in CMN form with the MIDI player as shown in Figure 6 after the user clicks the grey bar on the matched location as illustrated in Figure 5. Then, the user listens to the query melody with its score notation. If the user wants to modify the query, he first leftclick his mouse and then drag into the desired position on the music sheet applet. There are two buttons on the right of the generated notes; one is for listening to those scores and the other is for requerying. If the user wants to get a better result, just click the "Requery" button. Then, the system uses this modified query scores to reformulate the query.

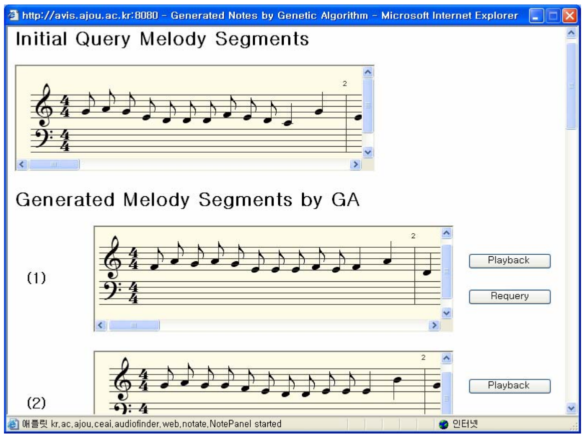

http://avis.ajou.ac.kr:8080 - Generated Notes by Genetic Algorithm - Microsoft Internet Explorer
Initial Query Melody Segments
Generated Melody Segments by GA
(1)
Playback
Requery
(2)
Playback
애플릿 kr.ac.ajou.ceai.audiofinder.web.notate.NotePanel started
인터넷

Figure 6. Feedback interface

4. EXPERIMENTS

In the experiment, user's vocal queries are captured on a microphone, and stored as PCM wave file with 8-bit, 22.05kHz, mono. We set framing length of 20ms as the minimum analyzable length..

For WAE method, we used the following parameters; global threshold of AE magnitude to 0.003, the unit window size to 16 frames, and the differential ratio to 20%. And for ADF threshold, we used the ratio of 50%. In the phase of classification of ADF onsets, we used the global threshold ratio as 33.3%. These values were observed as optimal from various experiments we performed.

Figure 7 shows the performance of our new method. It outperforms several previous methods. The missed onsets were reduced a lot when using the WAE. It is because the WAE contains the frames that AE could not produce. On the other side, without ADF, there existed many merged note segments among the missed onsets. Main role of ADF is splitting the merged note segment into repeated ones. Thus, the integration AE or WAE with ADF reduces the number of the missed onsets. The effectiveness of our method can be expressed in terms of recall and precision defined in (3).

$$\begin{cases} Re\,call = \dfrac{Detected\ onsets - False\ \ onsets}{Correct\ onsets} \\ Pr\,ecision = \dfrac{Detected\ onsets - False\ \ onsets}{Detected\ onsets} \end{cases} \qquad (3)$$

In Figure 8, we measured the relationship between the query length and the response time in each generation of our genetic algorithm. As we expected, a small number of generations with a few notes query such as 5 or 10 notes gives much better results than large number of generations with longer notes. We empirically found out that the optimal number of generation is 20 and reasonable query size is about 10.

5. CONCLUSIONS

In this paper, we have presented two new schemes for efficient music retrieval: (i) Extraction of pitch and duration information from user vocal query, and (ii) GA-based query reformulation to improve retrieval performance. We implemented a prototype system based on them and performed various experiments. The result showed that relevance feedback technique with genetic algorithm could achieve excellent retrieval performance.

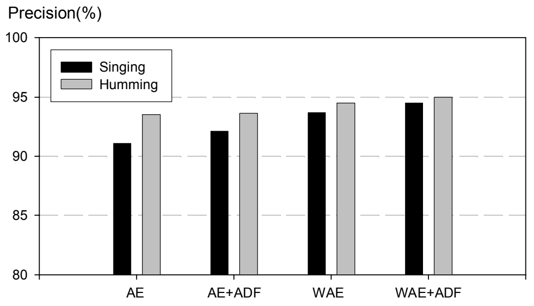


Figure 7. Effectiveness of ADF on AE/WAE

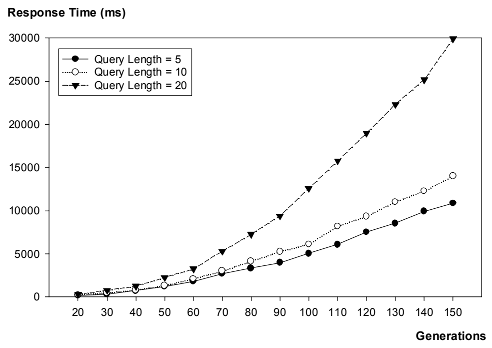


Figure 8. Efficiency of our algorithm

6. REFERENCES

[1] S. Rho and E. Hwang, "FMF: Query adaptive melody retrieval system," Journal of Systems and Software (JSS), Vol. 79(1), pp. 43–56, 2006.

[2] Y. Wang, Z. Liu, and J. Huang, "Multimedia Content Analysis using Both Audio and Visual Clues," Signal Processing Magazine, Vol.17, pp. 12 - 36, Nov. 2000.

[3] Stejic Z., Takama Y. and Hirota K., "Genetic algorithm-based relevance feedback for image retrieval using local similarity patterns," Elsevier Journal-Information Processing and Management, vol. 39, no. 1, pp. 1–23(23), January 2003.

[4] S. Park, S. Kim, K. Byeon, E. Hwang, "Automatic Voice Query Transformation for Query-by-Humming Systems," The conf. of (IMSA'2005), pp.197-202, Aug. 2005.

[5] Baeza-Yates, R., Ribeiro-Neto, B., "Modern Information Retrieval," Addison Wesley, 1999.