A LYRICS-MATCHING QBH SYSTEM FOR INTERACTIVE ENVIRONMENTS

Panagiotis Papiotis
Music Technology Group,
Universitat Pompeu Fabra
panos.papiotis@gmail.com

Hendrik Purwins
Music Technology Group,
Universitat Pompeu Fabra
hendrik.purwins@upf.edu

ABSTRACT

Query-by-Humming (QBH) is an increasingly prominent technology that allows users to browse through a song database by singing/humming a part of the song they wish to retrieve. Besides these cases, QBH can also be used to track the performance of a user in applications such as Score Alignment and Real-Time Accompaniment. In this paper we present an online QBH algorithm for audio recordings of singing voice, which uses a multi-similarity measurement approach to pinpoint the location of a query within a musical piece taking into account the pitch contour, phonetic content and RMS energy envelope. Experiments show that our approach can achieve 75.4% Top-1 accuracy in locating an exact melody from the whole song, and 57.8% Top-1 accuracy in locating the phrase that contains the exact lyrics – an improvement of 170% over the basic pitch contour method. Average query duration is 6 seconds while average runtime in MATLAB is 0.8 times the duration of the query.

1. INTRODUCTION

1.1 Presentation of the context

Query by Humming has gained attention as an approach, partly due to the increasing size of music collections; it is far easier to hum/sing the main melody for the song one wants to retrieve than search for it using the title and/or semantic labels. Further signs of the growing presence of QBH as an audio querying concept are also demonstrated by its inclusion as part of the of the MIREX contests since 2006.

However, this is not the only occasion on which QBH can be applied. Real-time Accompaniment systems such as Score Following/Alignment [1] attempt to align an existing score to the performance of a soloist in an online manner, very much like a human accompanist would align and adapt his/her performance to match that of the soloist.

Unfortunately, capable accompanists are hard to come by, especially for pieces of advanced difficulty. For this reason, an intelligent accompaniment system would be very useful for the soloist, as it would allow him/her to train and prepare for a performance without the need of a compromise for a sub-standard accompanist or the reduction of his opportunities to practice for a challenging piece.

In this context, a capable QBH system would be valuable as a front-end; it would automatically locate the soloist's position within the musical piece and provide the starting point for the alignment process. The soloist could stop and start again from a different point in the piece, without having to manually adjust the starting position.

However, virtually every musical piece contains repetitions of the same melody; simply put, there are several starting points for most queried melodies, a detail that is overlooked in most QBH implementations available currently. In the special case of the singing voice, a useful feature that could be utilized to discriminate between identical melodies are the piece's lyrics; with the exception of repeating parts such as the bridge or the chorus, every melodic line is coupled with a different lyric line. Therefore, if one can match the lyrics of the query to the lyrics of the reference, the exact location of the soloist within the piece would be pinpointed with perfect accuracy.

1.2 Related Work

As mentioned before, there is a significant amount of research done in the field of QBH, and as a standalone research field it is increasing in maturity. Early works on content-based audio or music retrieval are primarily based on signal processing and the acoustic similarity of the whole waveform [2].

Recent advances in the field utilize only the pitch contour of the query, which is directly transcribed from audio and compared to MIDI representations of all pieces within a database [3]. This approach yields satisfactory results, but strongly depends on the quality and accuracy of the query. Furthermore, this method performs a certain simplification over the input data to the point where discrimination between two or more candidates becomes a very hard task. Other approaches include a further range of features to calculate similarity, such as rhythm and pitch intervals [4], or relative interval slopes [5].

Predominantly, two different distance metrics are used in order to calculate the similarity between the query and the musical pieces within the database: *frame-based* and *note-based* similarity. Either one has its advantages; frame-base similarity is more accurate and ro-

bust, but is time-consuming. On the other hand, note-based similarity is faster, but offers less precision. A more efficient approach which utilizes the second metric can be found in [6], where the query is transcribed into pitch vectors and a list of candidate melodies is retrieved from the song database using *locality sensitive hashing*.

Another interesting approach from which our work borrows elements is the use of *multi-similarity fusion*, or the combination of the two distance metrics [7]; first, note-based similarity is used in order to quickly filter out the least similar candidates, and then frame-based similarity is applied to more accurately retrieve the best candidates.

Regarding lyrics recognition, a promising approach that is partly similar to our work approach is presented in [8], where a supervised Hidden Markov Model is used to recognize phonemes in a song using a pre-processed lyrics file, with an interesting application in QBH which achieves an accuracy of 57%. Another approach can be seen in [9], where an existing lyrics file is aligned to a vocal performance in Cantonese, using a combination of melody transcription, onset detection and Dynamic Time Warping (DTW) [10].

1.3 Our approach

Since we are trying to locate the exact position of a query within a single musical piece, the conditions and goals are relatively different to most of the cases presented above. Furthermore, the system has to work in a real-time accompaniment context; this restricts the average duration of the queries, since the QBH algorithm has but a small amount of seconds to return the located phrase.

Another goal of this work is to reduce the number of dependencies in terms of input as much as possible; for this reason, we avoided the use of auxiliary MIDI scores for the reference vocals as well as text files containing the lyrics for each phrase. This way, the only prerequisite for this system is a relatively stable audio recording of the reference vocals, such as the vocals in the originally recorded track. This recording is used to match the position of the queries sung by the user; it also serves as a reference through which the user's deviations in time and dynamics can be calculated to align the accompaniment to the user's performance. However, the latter part is still in progress and will not be discussed in this article.

The remainder of this paper is organized as follows: In Section 2, an overview of our system is provided. Section 3 focuses on implementation details for our approach. Section 4 shows our experimental results on the accuracy of the algorithm, and finally Section 5 contains our conclusions and future work recommendations.

2. SYSTEM OVERVIEW

As you can see in Figure 1, our system can be analyzed in four main processing modules. One is the pitch contour post-processing module, and three separate implementations of the Dynamic Time Warping algorithm –

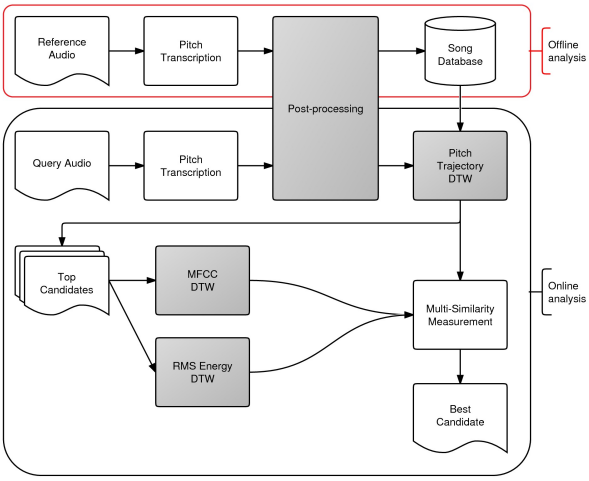

Figure 1. Overview of our system's four modules organized by online and offline analysis.

for the Pitch Contour, Mel-frequency Cepstral Coefficients and RMS Energy respectively. The two basic inputs for the system are the audio recordings of the reference vocals and the query.

2.1 Pitch transcription and post-processing

Both the reference vocals and the query are transcribed with the Yin algorithm [6], which produces a fairly accurate preliminary form of the F0 contour. However, YIN introduces several errors, the most prominent of which are the so-called "octave errors" – falsely choosing a fundamental frequency of twice or half the correct F0.

In order to overcome this problem, we first determine the tonal range of the recording by finding its maximum and minimum value where the aperiodicity of the signal is lower than a given threshold. Knowing the tonal range, we restore the values that are outside it by adding or subtracting a constant number. This way, the contour of the melody remains intact and is just "moved" using a certain offset.

Another problem we had to overcome are points in the recording containing consonants, roughness in the voice, or any brief burst of noise that "pollutes" the melodic content of the recording. Since these points do not have pitch, they can be removed using an aperiodicity threshold over which all values are set to zero. The gaps created are then bridged using an average value.

Finally, to smooth out the curve and speed up the DTW algorithm, we downsample the contour by a factor of 100.

2.2 Contour-based DTW

After obtaining the processed pitch contour of the query and the reference melody, we perform the DTW algorithm between the query and a sliding window of the reference that is equal to the length of the query. This returns a curve with the warping cost for every window of the reference (see Fig.2).

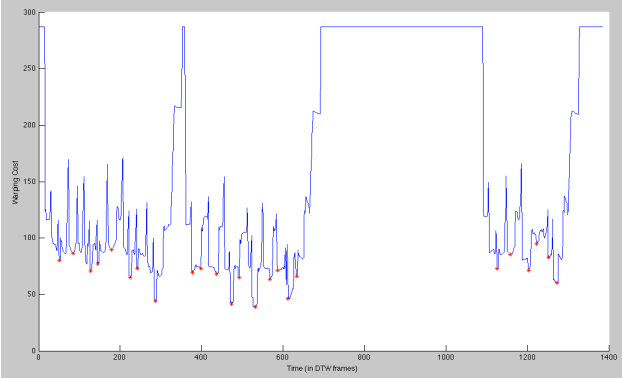

Figure 2. Contour-based DTW costs.

As seen in Figure 2, we select the local minima of this curve as the best candidates for the query. It can be observed that there are at least four phrases with the same melodic contour as the query; this is normal as songs have repeating melodies with different lyrics each time.

2.3 MFCC and RMS energy-based DTW

Having selected the best candidates, we try to match the phonetic content of the query based on the Mel frequency Cepstral coefficients and the pattern of words and silences as represented in the RMS Energy envelope. The DTW algorithm is performed between the retrieved candidates and the query twice more – once for the MFCCs and once for the RMS envelopes, thus producing two sets of warping costs.

2.4 Multi-Similarity Fusion

Finally, all three warping costs are combined, to help determine the best candidate. Each one of the cost vectors is normalized by its norm and added to the final costs vector. The minimum value of this vector is returned, signifying the position where the query was located.

3. IMPLEMENTATION DETAILS

The implementation of the proposed system was done in MATLAB. For the reference melodies, 7 vocal monophonic recordings of songs from the pop/rock genre were used, while the queries were recorded independently by a single user and comprised of 114 phrases with an average duration of 6 seconds each.

The whole system consists of two basic functions: *yinToTrajectory*, which performs the post-processing algorithm described in 2.1, and *QBHLyricFinder,* which calculates the DTW costs as shown in 2.2. In *yinToTrajectory*, the aperiodicity threshold is required as a parameter; values higher than the threshold are removed and replaced with an average value curve. In *QBHLyricFinder*, only the hop-size for the Contour-based DTW has to be adjusted by the user; it has been observed, however, that a hop-size of 400ms or less achieves the best results.

For the pitch contour & RMS energy DTW, Euclidian distance was used in order to construct the similarity matrix. For the Mel-frequency Cepstral coefficients, the cosine similarity between the two MFCC matrices was used, as is also the case with the DTW implementation found in [10].

As we need the system to work as part of an Interactive accompaniment application, computational efficiency is very important: after the user has sung the query, he/she keeps singing; the system must calculate the query's exact position within the reference and start playback from the point the user has reached by that moment. In average, our system's response time is 4.65 seconds for a 6.45-second query. Of course, the response might vary according to the number of candidates chosen during the pitch-contour DTW calculation. Out of these 4.65 seconds, 3.17 correspond to the YIN analysis of the query as well as the post-processing, 1.03 seconds correspond to the Contour-base DTW, and 0.47 seconds to the rest of the algorithm.

4. EXPERIMENTAL RESULTS

As mentioned before, 114 recorded phrases covering the whole of 7 different tracks were used as queries. In our context, a phrase is defined as a small group of words, matched with an individual melody, that stands as a conceptually distinct unit within the song – which is usually a line of the lyrics with its associated melody. Only the best-matching phrase is retrieved by the algorithm; we considered the output of the algorithm a hit, if the phrase returned had an overlap of at least 50% with the query.

Besides the main accuracy of the algorithm, we also calculated for each track the random guess accuracy for lyrics matching, the mean MFCC similarity between the reference and querying voice, the melodic variation of the track and the accuracy of the post-processed pitch contour.

4.1 Random guess accuracy

Since an overlap of at least 50% between the retrieved phrase and the query is considered a hit, the first frame of the retrieved phrase must be located at half the query's length before or after the actual first frame within the reference; more simply put, the overlap between the retrieved phrase and the query can either occur at the query's first or second half, but the duration of the retrieved phrase is always equal in length to the query. Therefore, the range of positions (in number of frames) that are considered correct is equal to

$$f_{query} = \frac{l_{query}}{h}, \quad (1)$$

where l_{query} is the length of the query and h is the hop size of the sliding DTW window, in frames. Similarly, the last frame of the retrieved phrase must be located at half the query's length before or after the actual last frame within the reference, so the range of all possible positions is equal to

$$f_{ref} = \frac{l_{ref} - l_{query}}{h}, \quad (2)$$

where l_{ref} is the length of the reference in frames.

This way, the random guess accuracy can be computed using the following formula:

$$acc = \frac{f_{query}}{f_{ref}} = \frac{l_{query}}{l_{ref} - l_{query}} \quad (3)$$

Some of the songs contain phrases that are repeated throughout its duration, therefore increasing the random guess accuracy. Moreover, when two identical phrases appear sequentially (i.e. the end of the first coincides with the beginning of the second), any frame between the middle of the first repetition and the middle of the second repetition is considered a hit. For these cases, the random guess accuracy is equal to

$$acc_2 = \frac{l_{query} * \left[n_r - (\frac{n_b}{2}) \right]}{l_{ref} - l_{query}}, \quad (4)$$

where n_r is the number of repetitions for that phrase and n_b is the number of shared boundaries between sequential phrases. Since in these cases the accuracy changes according to n_r and n_b, we compute it as a weighted average of all phrases within the song.

It can be argued that two identical phrases, which contain the same lyrics content and melody, might be emphasized differently in each repetition and can therefore qualify as separate phrases; this is currently viewed as a very subtle difference by our approach and such phrases are not treated individually. However, it is a valid case in some types of music (such as in operatic arias) and shall be investigated in the near future.

The overall random guess accuracy for each song as well as the average random accuracy can be seen in Table 1:

Song name	**Random guess accuracy**
She's leaving home	0.076
Butterflies & hurricanes	0.055
Nude	0.036
Bohemian Rhapsody	0.049
A day in the life	0.041
All the small things	0.134
Message in a bottle	0.066
Average baseline accuracy	**0.0471**

Table 1. Individual and average baseline accuracy for lyrics matching.

The average baseline was computed using a weighted sum, according to the number of queries for each song.

4.2 Average accuracy

The average accuracy of our algorithm was calculated as the number of queries located correctly over the total number of queries. In order to evaluate our results clearly and draw conclusions, we also calculated a number of features for each song, which are shown together with the average accuracy in Table 2.

Song ID	**Accuracy**	**Timbre similarity**	**Contour accuracy**	**Melodic variation**
SLH	0.61	0.3505	4	0.23
B&H	0.66	0.4091	4	0.27
N	0.61	0.7506	5	0.61
BR	0.57	0.374	1	0.57
ADITL	0.45	0.6172	2	0.29
ATST	0.6	0.3564	2	0.5
MIAB	0.6	0.2992	2	0.8
Overall	**0.585**			

Table 2. Accuracy and computed features (song IDs are derived from the initials of the song title).

Timbre similarity was calculated as the mean cosine similarity between a query and the relevant phrase from the reference recording, in order to observe how different singers (each with his/her own pronunciation and timbre) influence the lyrics matching.

Pitch contour accuracy was qualitatively graded from 1 to 5, according to the smoothness of the reference pitch contour as well as its similarity with the actual vocal line - it was observed that the pitch contour retains errors and noisy elements even after the post-processing.

Finally, melodic variation was calculated for each song as the number of unique phrases within a song over the total number of phrases in it; a melody is considered unique if its pitch contour is not repeated within the song. High melodic variation characterizes a piece where the vocal melodies are seldom reused, whereas low melodic variation characterizes pieces that feature repetitive melodies.

As our results show, average accuracy for our algorithm is 58.5% with a random guess accuracy of 4.7%, while the accuracy of our program when trying to only locate a phrase with the same contour is 75.4%. We also tested an implementation of the basic QBH algorithm using only the pitch contour to match the queried phrase; the accuracy amounted to 34% when trying to locate a phrase with the same lyrics, and 72.8% when trying to only locate a phrase with the same contour; this demonstrates that using other features besides pitch contour can actually increase the retrieval accuracy even when the objective is not to retrieve phonetic-matching content.

It can be seen from Table 2 that the most evident factor affecting the accuracy is the quality of the pitch transcription (*Contour accuracy*), although the calculation of this feature was qualitative. This is expected, since the performance of the MFCC-based matching is heavily improved when the candidates are fewer and more accurate. Since the timbre of the reference voice is statistically bound to be rather dissimilar to the querying voice, the number of candidates that 'survive' through the Contour-based DTW must be restricted only to the best-matching contours. The melodic variation does not have a big impact on accuracy since, based on our qualitative observations, almost all variations of a queried melody appear among the chosen candidates.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a lyrics-matching Query by Humming algorithm that can be used as a front-end for an interactive real-time accompaniment system specialized for singing voice. An audiovisual demonstration of our system is available on the Internet for evaluation purposes, on the address provided in [11]. Our experiments demonstrate that this approach shows promising results in the context of a time critical, single-output deterministic system. However, our experiments are limited in number and were tested with a single user; efforts to remediate this are currently underway.

An immediate improvement over the algorithm would be a better pitch contour post-processing module for the reference vocal recording, as it is demonstrated that its performance directly influences the accuracy; such an improvement could be the addition of an HMM-based model that would align an existing MIDI score to the reference recording, in order to avoid discontinuities and errors in the reference contour.

Another improvement that would bridge the gap between the contour-matching and the lyrics-matching accuracy would be to utilize the residual part of both the query and the reference recordings, in order to rectify the MFCC-based similarity measure. This way, all melodic content would be discarded when trying to match the lyrics in two phrases with an identical melody.

Finally, as this system is designed to be executed in an interactive environment, a mapping could be gradually performed between the reference recording and the user's voice, in order to increase the lyrics-matching accuracy through extended use.

6. ACKNOWLEDGEMENTS

The second author (HP) holds a Juan de la Cierva scholarship of the Spanish Ministry of Science and Innovation.

7. REFERENCES

[1] A. Cont: "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music", *Proceedings of International Computer Music Conference (ICMC)*, August 2008, Belfast, Ireland.

[2] J. T. Foote: "Content-Based Retrieval of Music and Audio." *In C.-C. J. Kuo et al, editor, Multimedia Storage and Archiving System II, Proceedings of SPIE*, volume 3229, pp.138—147,1997.

[3] A. Ghias, J. Logan, D. Chamberlin, B.C. Smith: "Query By Humming - Musical Information Retrieval in An Audio Database", *Proceedings of the third ACM international conference on Multimedia*, pp. 231—236, 1995.

[4] R.J. McNab et al: "Towards the Digital Music Library: Tune Retrieval from Acoustic Input". *Proceedings of Digital Libraries*, pp 11—18, 1996.

[5] A.L.P. Chen, M. Chang and J. Chen: "Query by Music Segments: An Efficient Approach for Song Retrieval". *Proceedings of IEEE International Conference on Multimedia and Expo*, 2000

[6] M. Ryynanen and A. Klapuri: "Query by humming of MIDI and audio using locality sensitive hashing". *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp 2249—2252, Las Vegas, Nevada, USA, Apr. 2008.

[7] 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", *IEEE International Conference on Audio, Language and Image Processing*, Shanghai, 2008.

[8] A. Mesaros, T. Virtanen: "Automatic Recognition of lyrics in singing", *EURASIP Journal on Audio, Speech, and Music Processing vol. 2010*, Article ID 546047, 11 pages, 2010.

[9] C.H. Wong, W.M. Szeto, K.H. Wong: "Automatic lyrics alignment for Cantonese popular Music", *Multimedia Systems*, vols. 4-5, no. 12, pp. 307—323, 2007.

[10] D. J. Berndt, J. Clifford: "Using dynamic time warping to find patterns in time series", *KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, pp. 359—370, Seattle, 1994.

[11] D. Ellis: "Dynamic Time Warping in MATLAB", *http://www.ee.columbia.edu/~dpwe/resources/matlab/dtw/*, retrieved on April 2010.

[12] http://www.youtube.com/user/qbhlyricsalignment