Real-time accompaniment using lyrics-matching Query-by-Humming (QBH)

Panagiotis Papiotis

MASTER THESIS UPF / 2010

Master in Sound and Music Computing

Master thesis supervisor:

Hendrik Purwins

Department of Information and Communication Technologies

Universitat Pompeu Fabra, Barcelona

UNIVERSITAT POMPEU FABRA

Abstract

We present an automatic accompaniment system for the singing voice, which allows the user to begin singing from any point within the piece he/she desires. The system is based on the Dynamic Time Warping (DTW) algorithm, which is used as both a similarity measure as well as an alignment tool between the live performance and the reference. The user's position within the piece is initially determined with a variation of current Query by humming/singing methods, based on three sets of acoustic features; pitch trajectory, RMS energy envelope and MFCCs. After locating the user, the system adapts the tempo of the accompaniment utilizing a known on-line DTW method. The system is implemented in MATLAB and Java, while audio playback is handled using Max/MSP.

Acknowledgments

I would like to thank my supervisor, Hendrik Purwins, for his invaluable help and understanding throughout this year, as well as his keen ability to motivate and guide me, irrelevant of the subject.

Xavier Serra for giving me the opportunity to participate in this research community and become a member of the MTG, as well as everyone else who either taught me or assisted me during this master's course - it is often that you find your way without even looking for it.

Simon Dixon for providing me with the MATCH source code, which proved an integral part of this work.

My colleagues and friends Ahmed Nagi, Panagiotis Melidis, Felix Tutzer as well as Marco Marchini for the discussions and experiences with them that shaped this Thesis to its current form.

My girlfriend Elena for her patience and understanding.

Finally, my parents and sister, for their unquestionable support in every way possible; this thesis is dedicated to and exists because of them.

Contents

List of figures

List of tables

1. Introduction

In this chapter we provide a brief introduction of the concepts this thesis is built around. First, the context and motivation of our research is presented, along with the final scope of our work. Then, the state of the art is presented in two parts, regarding the two research fields this thesis is based upon: Query-by-Humming, and Score Following.

1.1 Presentation of the context

The notion of accompaniment, as a musical part that supports or partners a solo instrument, voice, or group, is central to music performance. Most forms of music are based on the same formula, i.e. a melodic part evolving over an accompaniment part, even if the roles of the accompanist and the soloist are interchangeable (as in the case of call-and-response music, or dialogue accompaniment). Moreover, the same person often handles both roles; in solo piano pieces, for example, one hand (traditionally the left hand) carries the accompaniment, while the other performs the melodic part.

With the exclusion of solo musical pieces, where the musician can practice, perfect, and perform a musical piece on his/her own, an accompanist is needed in order to achieve these three tasks; and therein lies a very important problem in the musical world: *where can I find a capable and willing accompanist?*

The primary and most important goal of the accompanist is to follow the soloist; this is not an easy task, especially in advanced pieces where the difficulty level of the accompaniment is equal to, or even greater than that of the solo part. The accompanist must keep the metrical and rhythmic structure, which can be especially hard in cases where the solo part contradicts that structure on purpose; follow subtle and not-so-subtle changes in tempo and dynamics dictated by the soloist; and, quite often, communicate with the soloist by the means of nods or gestures, in order to coordinate for events such as simultaneous notes, musical 'breaths', and changes in tempo (accelerando/ritenuto).

Of course, the task of accompanying can be as enjoyable as that of leading, particularly during the final performance. However, during practice, the accompanist is expected to know the piece beforehand, and follow the directions of the soloist as to which part is to be practiced, how many times, et cetera. For this reason, capable accompanists are very hard to come by, and soloists often have to travel in order to practice with their accompanist of choice, or vice versa.

In this context, automatic accompaniment is a task with immediate applications. Besides contemporary classical music and modern music, where tape accompaniments and loops have already become a widespread approach, automatic accompaniment is

also eminent in karaoke machines and music videogames (such as SingStar, Rock Band etc).

1.2 Scope and orientation

The main goal of the present work is to create a computer-assisted accompaniment system for the singing voice, which will allow singers to practice on a particular piece. As it will be seen in the following section, performance-tracking systems for monophonic instruments already exist. However, there is an important difference between the singing voice and other monophonic instruments – the lyrics content. Pieces for singing voice are usually based on repetitions of the same melody, while the lyrics change – this is particularly the case in popular music genres such as pop, rock, jazz, et cetera.

Since this system was intended for practicing, we considered this characteristic of the singing voice an important factor; most pieces usually consist of alternations between verses and choruses, each of which has a more or less fixed melodic line. Therefore, it was important that the system could differentiate between two points in the piece that have the same melody, in order to give the user the choice to start practicing from any possible verse, or lyric.

To this end, our work got divided in two separate fields:

- that of **Query by Humming/Singing**, where the system searches inside the piece to initially determine the user's position based on the melodic & lyrics content, and
- that of **Score Following / Performance tracking**, where the system continuously tracks the user's position, adjusting the tempo of the accompaniment.

It must be noted that throughout the project's evolution, Query by Humming (which will from now on referred to as QBH) became a much more challenging and central to this work part than was initially expected. Therefore, the main contributions of this work lie in the field of QBH moreso than Score Following.

1.3 State of the Art review

1.3.1 Query by Humming

Query by humming (QbH) is a music retrieval system that involves taking a user-hummed melody (input query) and comparing it to an existing database. The system then returns a ranked list of music closest to the input query. Lately, QBH has gained widespread 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.

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, such as J.T. Foote's work (Foote, 1997), are primarily based on signal processing and the acoustic similarity of the whole waveform.

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. While this approach yields satisfactory results, it strongly depends on the quality(Ghias, Logan, Chamberlin, & Smith, 1995) and accuracy of the query, as well as the melodic transcription module that is used. Furthermore, in order to reduce computing costs, 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. Efforts to remediate this problem include a further range of features to calculate similarity, such as rhythm and pitch intervals (McNab, 1996), or relative interval slopes (Chen, Chang, & Chen, 2000).

It must be noted that although the concept of a QBH system is relatively straightforward and simple to grasp, the inner workings, algorithms and data structures used in every approach are often very complicated. However, they usually concern the querying stage of the algorithm, where the query is being compared to the contents of the database to retrieve the best results; since our approach does not feature such a stage, we will not mention that stage of the algorithm further on.

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. In frame-based similarity, the query representation has a time resolution analogous to that of the FFT frames, and is therefore more detailed. In note-based similarity, the query is transcribed and parsed in note segments, thus representing the query in a string of events.

Either one has its advantages; frame-base similarity measures are more accurate and robust, but the similarity calculation 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 (Ryynanen & Klapuri, 2008), 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 (Wang, Huang, Hu, Liang, & Xu, 2008). At first, note-based similarity using the Earth Mover's Distance (EMD) algorithm is used to find the top 20% candidates from the MIDI song database. Then, frame-based similarity using the Dynamic Time Warping (DTW) (Berndt & Clifford, 1994),(Ellis, 2010) algorithm is applied, to rank these

candidates more accurately. This method has the following advantage: irrelevant songs are quickly discarded using the fast, note-based scoring algorithm. It is then computationally viable to use frame-based similarity to obtain more accurate results. The success of this approach is demonstrated by the fact that the algorithm finished first at the MIREX 2008 QBH task, with a real-time ratio of 13.9%.

In the research field of lyrics recognition for QBH purposes, there is still ample room for research. Among existing research is the work presented in (Wong, Szeto, & Wong, 2007), where the lyrics of Cantonese popular songs are aligned to the recording using DTW. Firstly, the vocal part in commercial CD recordings is enhanced to estimate the pure vocal signals using a source separation algorithm. Then, the onsets of the characters sung (retrieved from a text version of the lyrics) are detected using a relative difference onset detection method. Since many non-vocal onsets are detected, they are pruned by the singing voice detection classifier, which classifies onsets into vocal and non-vocal. After that, a set of features (relative pitch, time distance) is extracted from the lyrics and the audio signal. Finally, the start time and the end time of each lyrics sentence are obtained by the dynamic time warping algorithm.

A more relevant to our work project can be found in (Mesaros & Virtanen, 2010), where speech recognition methods are applied directly on singing voice audio to extract the lyrics. Phonemes and words are extracted from monophonic audio recordings, as well as polyphonic recordings with the aid of a source separation algorithm. Due to the lack of large enough singing databases to train a singing recognizer, a phonetic recognizer that was trained on speech was used, applying speaker adaptation techniques to adapt the models to singing voice. Phoneme and word recognition is achieved through the use of a hidden Markov model, trained on speech database text and a database of textual lyrics for the phoneme and word-level, respectively. The algorithms are tested on two different applications: a speech recognizer, where textual lyrics are aligned to vocals in polyphonic music, obtaining an average error of 0.94 seconds for line-level alignment, and a query-by-singing retrieval application, with a Top-1 accuracy of 57%.

1.3.2 Score following

Score Following is the task of tracking a musician's position within a piece, in reference to the formal score notation of that piece. Through this tracking procedure, an alignment between the performance and the score can be performed; thus adjusting the objective, inexpressive score to the expressive performance of a musician.

This procedure (which can be performed both online and offline) can be of use in several scenarios; the most evident of which is automatic accompaniment. It is also often used for triggering visual and other events during live performances, automatic page turning (Arzt, 2007), and performance scoring (Dixon, An on-line time warping algorithm for tracking musical performances, 2005).

One of the pioneers in the field of Score Following is Roger Dannenberg, whose ongoing work since 1984 has greatly influenced existing research (Dannenberg, An on-

line algorithm for real-time accompaniment., 1984). Dannenberg's approach initially featured a dynamic programming algorithm, which tries to calculate the best match between a MIDI score of the performance, and the pitch transcription of the performance itself. Further improvements to this algorithm feature support for polyphonic instruments (such as the piano), detection and matching of musical ornamentations (such as glissandi, trills, etc) using different matching algorithms, and the implementations of multiple matchers running at different locations.

Another important piece of research is Vercoe's "Synthetic Performer" (Vercoe, 1984), presented in 1984 as well. Vercoe's system utilizes both pitch information as well as fingering information from the flute, and functions as an automatic accompaniment application – adapting tempo, dynamics, and phrasing according to the real-time performance of the user.

Based on these two approaches, several variations of the matching algorithms & transcription methods have been implemented. Some based the matching algorithm on predefined phrases (Baird, Blevins, & Zahler, 1993), "skip lists" of notes to be matched/expected(Puckette, 1995), or temporal structure of notes (Vantomme, 1995).

A somewhat different model, which marked a new direction in Score Following, was the Stochastic Method presented by Grubb and Dannenberg (Grubb & Dannenberg, 1997). In this approach, the user's position is represented as a continuous probability density function over the score position in time. The area under this function between two score positions indicates the probability that the user is within that region of the score. The complete area of the PDF is always equal to 1, indicating it is 100% likely that the user is in the score. As the performance progresses, new observations are reported updating the PDF to describe the user's new position.

This approach, as well as the subsequent progression towards statistical models, demonstrates the fact that live performances often contain several errors as well as spontaneous artistic liberties taken by the performer. Since the performance is nearly never "perfect", a stochastic model is a very natural approach to cope with these uncertainties.

Another expansion towards the Stochastic Method was marked with Pedro Cano's HMM-based approach in (Cano, Loscos, & Bonada, 1999). In this approach, three left-to-right Hidden Markov Models are used to describe events; the note, no-note and silence model. *Note* and *No-Note* are modeled with three states each (attack, steady-state and release), representing notes and unpitched segments, while the *Silence* model has only one state. A set of descriptors derived from audio (such as Energy, Zero Crossing rate, F0 etc) are used as input to the system, and the alignment is achieved using the Viterbi algorithm.

Further research on the field led towards the use of Dynamic Time Warping (DTW), a method orginally used in speech recognition; in its essence, DTW is a method for aligning time series. However, the time complexity of the algorithm has, until recently, made it viable only for offline Score Following applications (Orio & Schwarz, 2001), although efforts have been made to reduce the runtime of the matcher

(Dannenberg & Hu, Polyphonic audio matching for score following and intelligent audio editors, 2003).

The only exception to the above is Simon Dixon's performance tracking system MATCH (Dixon, An on-line time warping algorithm for tracking musical performances, 2005), which is based on an on-line modification of the DTW algorithm that permits the system to compute the alignment between the reference and the user in real time. Since the second part of our work heavily relies on MATCH, the details of Dixon's system will be presented in the next chapter.

2. Dynamic Time Warping

In this chapter we provide a brief presentation of Dynamic Time Warping; it was deemed necessary to explain the basics of the algorithm, as it is central to both parts of our research.

2.1 The algorithm

Dynamic time warping (DTW) is a technique for aligning time series, with its main application being in the domain of speech recognition, for three decades (Berndt & Clifford, 1994). Although as a speech recognition technique it has been largely superseded by Hidden Markov Models (HMM), it remains very useful today, more so for its usefulness as a similarity measure.

DTW aligns time series $\{U = u_1, \ldots, u_m\}$ and $\{V = v_1, \ldots, v_n\}$ by finding a minimum cost path $W = W_1, \ldots, W_l$. Each W_k is an ordered pair (i_k, j_k), such that $(i, j) \in W$ and the points u_i and v_j are aligned. The alignment is assessed with respect to a local cost function $d_{U,V}(i, j)$, usually represented as an $m \times n$ matrix, which assigns a match cost for aligning each pair (u_i, v_j). The cost is 0 for a perfect match, and is otherwise positive. The path cost $D(W)$ is the sum of the local match costs along the path:

$$D(W) = \sum_{k=1}^{l} d_{U,V}(i_k, j_k) \tag{1}$$

Several constraints are placed on W, namely that the path must be:

- bounded by the ends of both sequences,
- monotonic, and
- continuous.

Other local path constraints are also common, which alter the monotonicity and continuity constraints to allow increments of up to two or three steps in either direction and/or require a minimum of at least one step in each direction. Additionally, global path constraints are often used, such as the Sakoe-Chiba bound, which constrains the path to lie within a fixed distance of the diagonal (typically 10% of the total length of

the time series), or the Itakura parallelogram, which bounds the path with a parallelogram whose long diagonal coincides with the diagonal of the cost matrix.

The minimum cost path is calculated with a time complexity of $O(n^4)$, using the following dynamic programming recursion:

$$D(i,j) = d(i,j) + \min \left\{ \begin{array}{l} D(i,j-1) \\ D(i-1,j) \\ D(i-1,j-1) \end{array} \right\} \quad (2)$$

,where `D(i,j)` is the cost of the minimum cost path from (1,1) to (i, j), and `D(1, 1) = d(1, 1)`. The path itself is obtained by tracing the recursion backwards from `D(m, n)`.

2.2 DTW as an alignment index

Figure 2.1 demonstrates the alignment path between two time series:

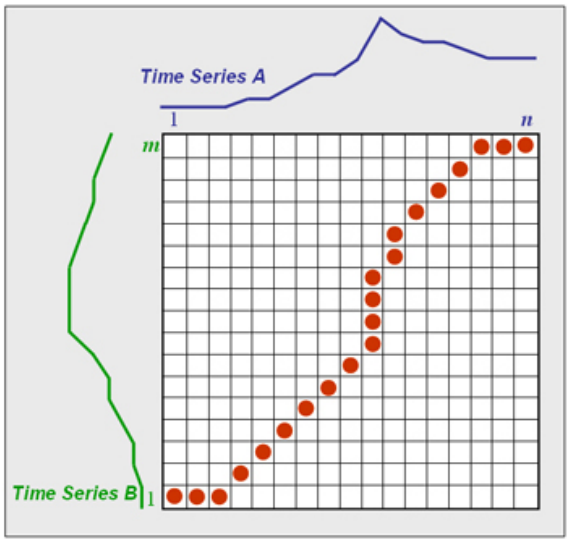

Figure 2. 1 - DTW path for the alignment of two time series

Each step in the path signifies the alignment between two points in the time series; if the path advances by a column, only time series A advances in time, therefore aligning the new point in the time series A to the same point in time series B as before. If the path advances by a row, the opposite happens, and time series B advances in time. If, finally, the path moves diagonally and therefore in both directions, both time series

are advanced by a point. Figure 2.2 visualizes the point-by-point alignment performed by the DTW algorithm:

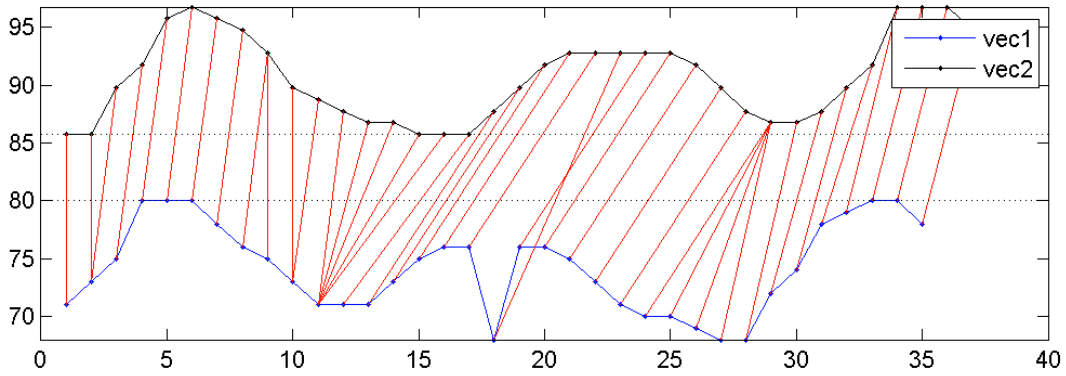

Figure 2. 2 - Point-by-point alignment for two time series

2.3 DTW as a similarity measure

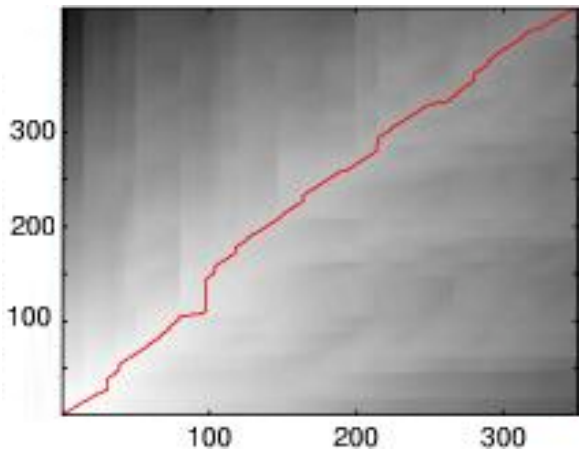

Figure 2. 3 - Cost matrix of the DTW algorithm

Since the cost of every alignment is added to the next point of the path, the final point of the alignment path contains the overall cost for warping the two time series. In this respect, DTW is also very useful as a similarity measure; in fact, it is the main similarity measure used in this work, as it allows for mostly accurate results even if one of the time series contains mistakes or inconsistencies.

This is demonstrated by Figure 2.3: each point in the cost matrix represents the alignment cost for that cell, ranging from light tones for low costs to darker ones. The final alignment path (in red) is seen ascending from the bottom left corner to the top right corner, passing through the points of lowest cost. The top right cell of the cost matrix represents the overall alignment cost.

3. Method Overview

In this chapter we provide an overview of the methods used in our system. First, we state the specific goals in our approach and how they differentiate this system from relative work. Then, our method for the lyrics-matching QBH algorithm is presented. Finally, we briefly describe Dixon's On-line Time Warping algorithm that was adopted by our system, and how we utilize it in our Score Alignment algorithm.

3.1 Our approach

Since we are trying to locate the exact position of a singer within a single musical piece, the conditions and goals are relatively different to most of the cases presented in section 1.3.1.

For example, 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 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 featured pieces as well as text files containing the lyrics for each phrase. This way, the system prerequisite is a relatively stable audio recording of the reference vocals, such as the vocals in the originally recorded track. This recording is used to locate 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.

3.2 Lyrics-matching QBH

As it can be seen in Figure 3.1, our lyrics-matching QBH 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 - 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.

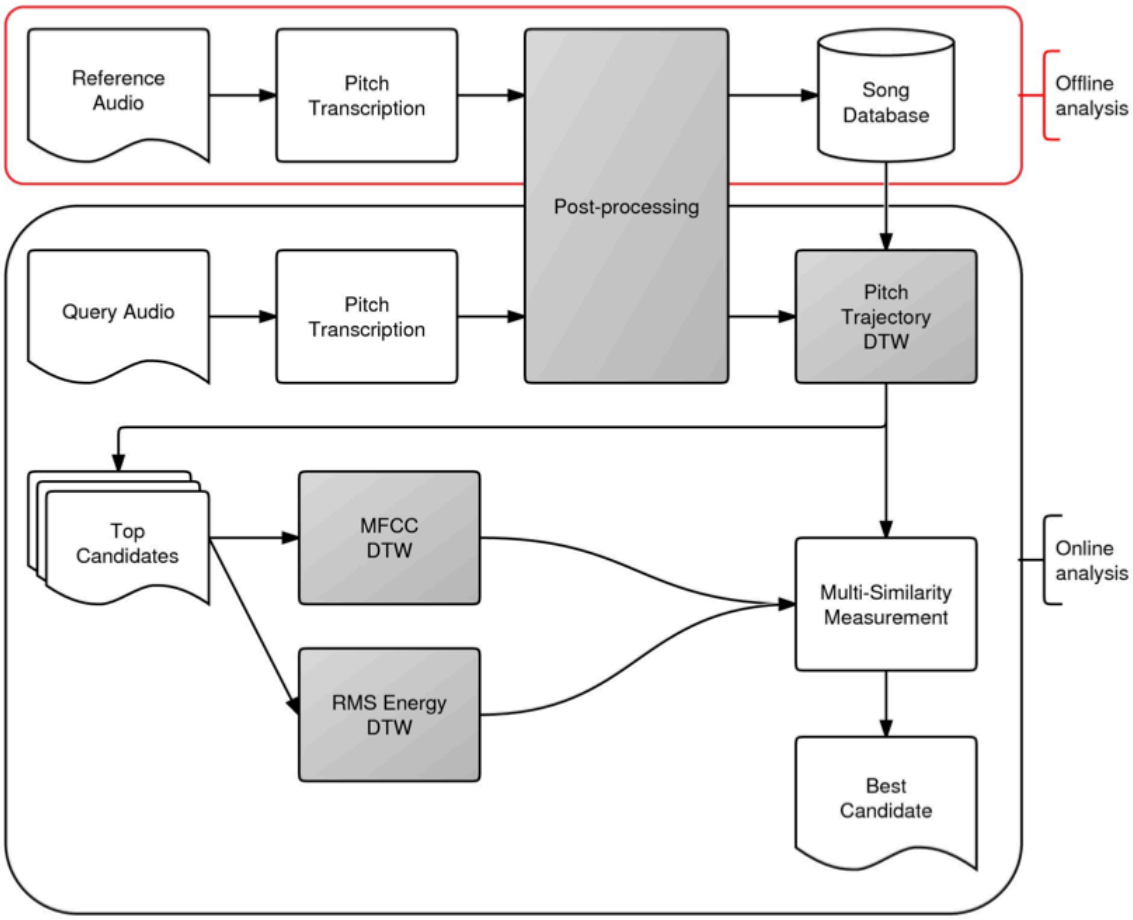

Figure 3. 1 - An overview of the lyrics-matching QBH system

A brief summary of the algorithm is the following:

- The pitch trajectory for the reference vocals is transcribed and post-processed beforehand, in order to speed up the algorithm.
- As soon as the user has finished singing the query, the system transcribes it using the same transcription and post-processing module.
- The transcribed query is then compared to the reference using the above representation frame-by-frame using three different DTW implementations, to obtain the best-matching position.

In the following section, each module is described in detail.

3.2.1 Pitch transcription and post-processing

Both the reference vocals and the query are transcribed using the Yin algorithm (de Chevegne & Kawahara, 2002), a pitch transcription system based on the Autocorrelation method. Although Yin produces a fairly accurate preliminary form of

the fundamental frequency (F0) contour, it also introduces several errors which have to be rectified. Of these errors, the most prominent are the so-called *Octave errors*, or choosing an F0 of double or half the actual frequency.

Our initial solution to the problem involved restraining the whole contour to the range of an octave; every value below or above that octave was shifted to belong in that interval. However, this led to the oversimplification of the incoming data, making it hard to distinguish between similar segments in the reference.

It quickly became obvious that this oversimplification was detrimental to the recall rate of the retrieval algorithm. Therefore, we opted for a milder interference: we first filter out the most "uncertain" points in the contour, by using a -rather strict- aperiodicity threshold of 0.008. This way, we obtain the accurate tonal range of the query, i.e. the highest and lowest pitch within it. Knowing the tonal range, we restore the values that are outside it by adding or subtracting a constant number that is multiple of an octave. This way, the contour of the melody remains intact and is just "moved" using a certain offset.

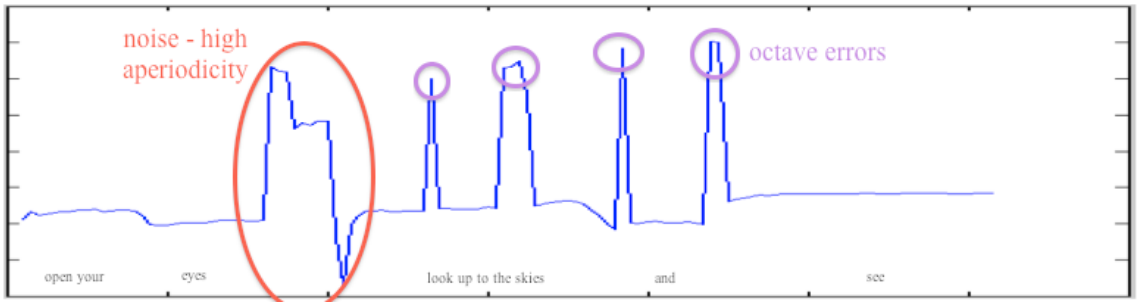

Figure 3. 2 - Transcribed pitch contour before the post-processing module

Another problem we had to overcome were points in the recording containing consonants, roughness in the voice, or any unvoiced sound 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, in order to increase the speed of the algorithm, the pitch contour is downsampled by a factor of 100. Figures 3.2 and 3.3 show the transcribed pitch contour before and after the post-processing module.

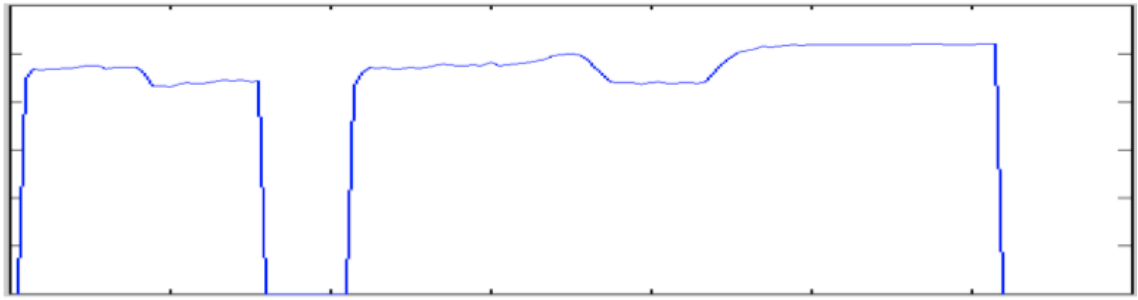

Figure 3. 3 - Transcribed pitch contour after the post-processing module

3.2.2 Pitch contour-based DTW

After obtaining the processed pitch contour of the query and the reference melody, we consecutively perform the DTW algorithm between the query and a sliding window of the reference that is equal to the length of the query; an example step from this can be observed in Figure 3.4.

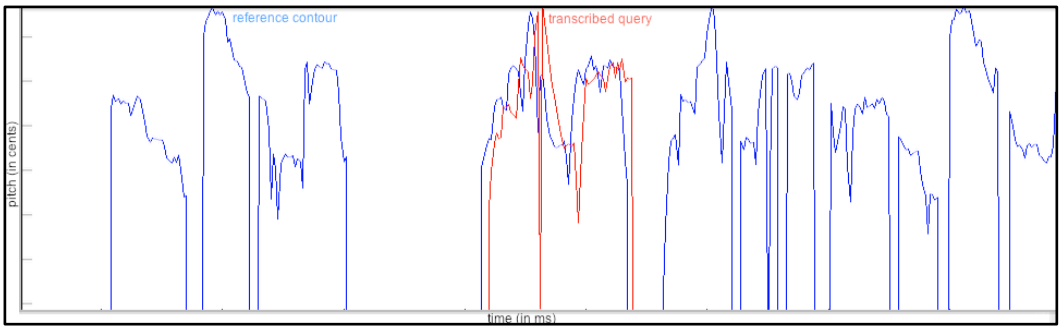

Figure 3. 4 - Example frame from the contour-based DTW

In the above figure, the query (seen here in red) is compared to a particular segment of the reference. In the next step of the algorithm, the query will be shifted forward in time by a given hopsize (we obtained optimal results using a hopsize of 400 milliseconds) and compared to the next corresponding segment of the reference.

Using this technique, we obtain a set of DTW alignment costs, for every frame of the matching algorithm explained above; points of low cost represent segments in the reference that are similar to the query. An example can be seen in Figure 3.5.

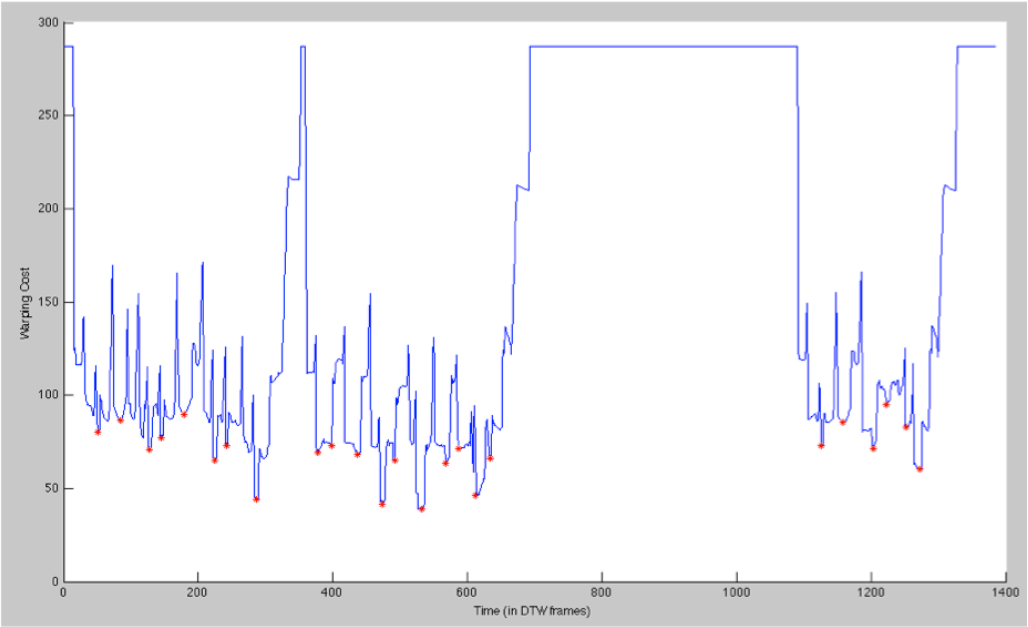

Figure 3. 5 - DTW alignment costs for the whole song

As seen in the above figure, 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 expected as songs have repeating melodies with different lyrics each time.

3.2.3 RMS Energy and MFCC-based DTW

Having filtered out most irrelevant phrases within the reference, we now need to locate the segment with the same lyrics content as the query, that is presumably among our picked candidates. In order to achieve this we extract two sets of descriptors; the RMS energy envelope, which represents the pattern of words and silences in the sung segment, and the Mel-Frequency Cepstral Coefficients (MFCCs) for 13 frequency bands, which represents the phonetic content.

In order to keep the runtime short as well as retain a good recall rate for our algorithm, we only extract these descriptors for the candidate segments chosen in the previous step. 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 more sets of alignment costs.

3.2.4 Multi-similarity fusion

Finally, having obtained three different sets of alignment costs between the query and the candidate segments of the reference, we combine all three costs in order to 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 as the output, signifying the temporal position within the reference where the query was located.

3.2.5 Implementation details

The implementation 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 3.2.1, and `QBHLyricFinder`, which calcu- lates the DTW costs as shown in 3.2.2-4. 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 hopsize for the Contour-based DTW has to be adjusted by the user.

For the pitch contour & RMS energy DTW, Euclidian distance was used in order to construct the similarity matrix. For the MFCCs, the cosine similarity between

the two MFCC matrices was used, as is also the case with the DTW implementation by Dan Ellis (Ellis, 2010).

As we need the algorithm 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. After calculating the starting point, we send it via an OSC message to a Max/MSP patch which handles all audio playback in our system.

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-based DTW, and 0.47 seconds to the rest of the algorithm.

3.3 Score Alignment using On-line Time Warping (OLTW)

After locating the user's position within the piece, the next step is to start aligning the tempo of the accompaniment according to the user's performance. A audio recording of the piece without the vocals is used as the accompaniment file; it can therefore be argued that this part of our system does not classify as Score Alignment, since there is no actual score of the performance.

In fact, Dixon's method (which our system builds upon) was introduced as a *Performance tracker*. This method is based on a novel, linear-time variation of the DTW algorithm called *On-line Time Warping* (OLTW) presented by Dixon in DAFx '05; a brief presentation of the algorithm is provided below, in order to provide a base on which our additions to the algorithm are demonstrated.

3.3.1 Performance tracking using On-line Time Warping

As in every variation on the DTW algorithm, a representation for the two time series is needed in order to calculate the alignment costs. Dixon uses a low-level spectral representation of the audio data which is generated from the STFT of the signal, using a hamming window of 46 ms, with a hop size of 20 ms. A Hamming window with the size of 46ms, and a hop size of 20ms is used.

The frequency axis was mapped to a scale which is linear at low frequencies and logarithmic at high frequencies; the lowest 34 FFT bins (up to 370Hz,) are mapped linearly, while the bins from 370Hz – 12.5kHz were mapped onto a logarithmic scale with semitone spacing by summing energy in each bin into the nearest semitone element. Finally, the remaining bins above 12.5kHz are summed into the last element

of the new scale. The resulting representation is a vector of 84 points per FFT frame. The final audio frame representation uses a half-wave rectified first order difference, so that only the increases in energy in each frequency bin are taken into account.

Euclidean distance is used to calculate the distance between feature vectors, and the cost matrix is calculated using the simple dynamic programming recursion shown in 2.1, equation (2).

The alignment path estimation function is particularly important, as it is the main point that sets this algorihtm apart from the traditional DTW approach. In standard DTW, the lengths of the sequences provide one of the boundary conditions for the search; in the on-line case, this boundary condition must be estimated along with the optimal path. A consequence of this is that the diagonal of the cost matrix is unknown, so the global path constraints cannot be directly implemented; for this reason, the OLTW algorithm calculates an *adaptive diagonal*, though which the complexity of the algorithm is reduced.

Moreover, since only one of the time series is fully known, global path constraints cannot be implemented directly. An incremental solution is required, so the minimum cost path must be calculated in the forward direction. Furthermore, in order to run in real time with arbitrarily long series, the complete algorithm must be linear in the length of the series, so that the incremental step is bounded by a constant.

Pseudocode for the OLTW algorithm (taken from (Dixon, An on-line time warping algorithm for tracking musical performances, 2005)) can be seen in Figure 3.6:

```
ALGORITHM On-Line Time Warping
   t := 1; j := 1
   previous := None
   INPUT u(t)
   EvaluatePathCost(t,j)
   LOOP
      IF GetInc(t,j) != Column
          t := t + 1
          INPUT u(t)
          FOR k := j - c + 1 TO j
              IF k > 0
                  EvaluatePathCost(t,k)
      IF GetInc(t,j) != Row
          j := j + 1
          FOR k := t - c + 1 TO t
              IF k > 0
                  EvaluatePathCost(k,j)
      IF GetInc(t,j) == previous
          runCount := runCount + 1
      ELSE
          runCount := 1
      IF GetInc(t,j) != Both
          previous := GetInc(t,j)
   END LOOP
```

```
FUNCTION GetInc(t,j)
   IF (t < c)
        return Both
   IF runCount > MaxRunCount
        IF previous == Row
           return Column
        ELSE
           return Row
   (x,y) := argmin(pathCost(k,l)), where
                (k == t) or (l == j)
   IF x < t
        return Row
   ELSE IF y < j
        return Column
   ELSE
        return Both
```

Figure 3. 6 - Pseudocode for the OLTW algorithm

Given that U is the partially unknown sequence, at each time t we seek the best alignment u_1 , ..., u_t to some initial subsequence of V . The user-defined parameter c determines the width of the search band, while he pointers t and j point to the current positions in U and V and are initialized to point to the start of each series.

In the main algorithm, the path cost matrix is consecutively calculated by partially expanding towards a row or a column; this decision is handled by the GetInc function. Finally, path costs for the c x c cost matrix are calculated.

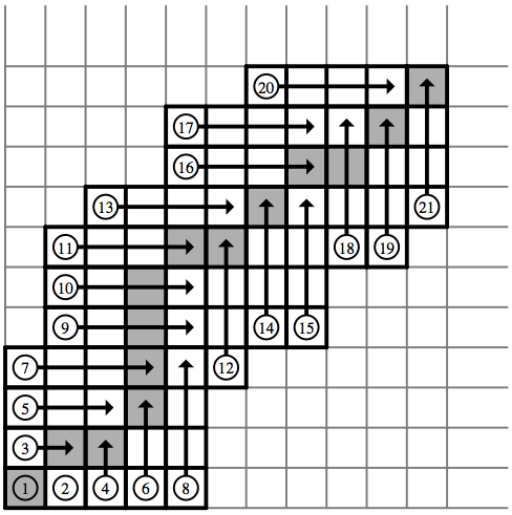

Figure 3. 7 - OLTW algorithm example (c=4)

In Figure 3.7, an example for the OLTW algorithm can be seen. The algorithm is run with a search width of 4. All calculated frames are framed in bold, while the optimal path is coloured grey.

For further information regarding the OLTW algorithm, we redirect the reader to (Dixon, An on-line time warping algorithm for tracking musical performances, 2005).

3.3.2 Tempo alignment

The forward path curve calculated by the OLTW algorithm provides a mapping between the partially unknown sound and the reference; therefore, the slope of this curve can be seen as the relative tempo between the two pieces. See Figure 3.8:

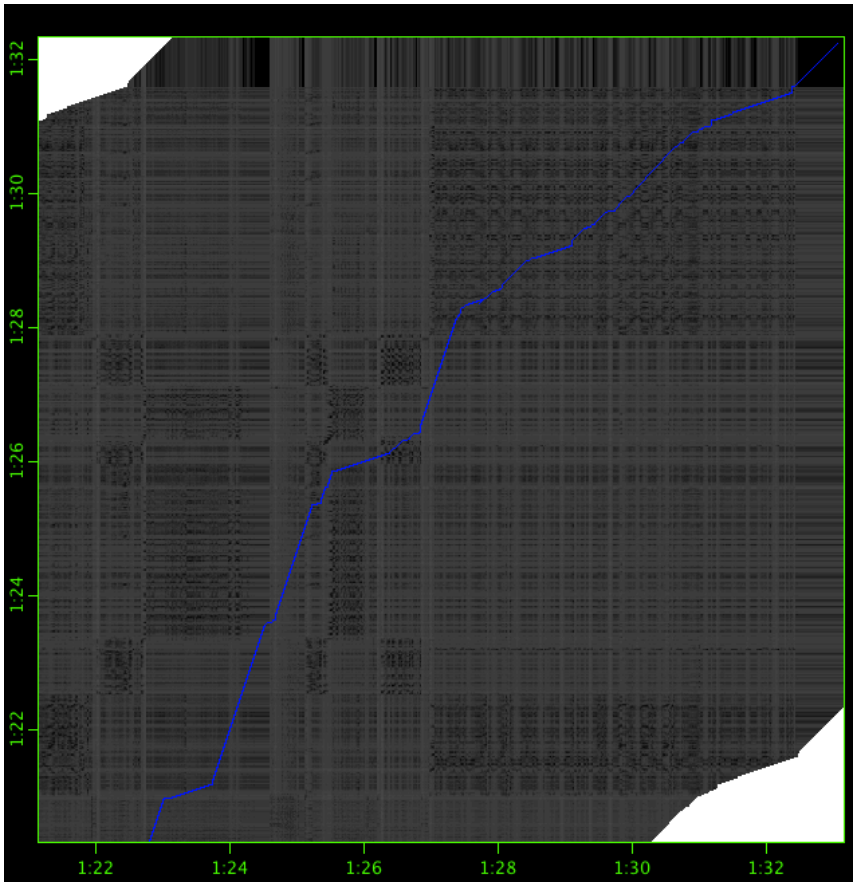

Figure 3. 8 - Screenshot from the execution of the OLTW algorithm

In the above figure, the forward path is shown in blue colour; the x axis represents the timeline for the partially known incoming live performance, while the y axis represents the timeline for the reference audio.

Our initial approach was to modify the playback speed of the accompaniment by multiplying it with the slope of the alignment curve. However, this suffered from the following problem: even though the slope is bounded between the values 0.33 and 3 so as to avoid abrupt or extreme tempo changes, it is possible that the slope comes close to these values, even if for a short time period (2-3 seconds). The resulting tempo change causes the accompaniment file to either advance too fast or fall behind quite a lot; after this, it was very hard for the accompaniment to "recover" and synchronise with the user.

For this reason, we adopted a constantly updating function which keeps track of two indices; the first index (`indexLive`) signifies the position of the live performance in relation to the reference, while the second index (`indexAcc`) signifies the current position of the accompaniment file. In order to reduce computational costs and facilitate

real-time execution, the above step is executed every 50 frames, or every second. The following pseudocode demonstrates how our algorithm works:

```
indexAcc = indexAcc+factor;
if (time passed since the last check == 50ms)
{
      indexLive = current relative position of the user within the piece;
      factor = slope of the alignment curve;
      boost = abs(indexLive-indexAcc)/500;
      if (indexLive > indexAcc)
      {
        //Accompaniment tempo must be increased
        factor = factor + boost;
      }
      else if (indexLive < indexAcc)
      {
        //Accompaniment tempo must be decreased
        factor = factor - boost;
      }
      set current tempo = factor;
}
```

Figure 3. 9 - Pseudocode for the tempo alignment method

As it can be seen, we derive the new tempo using the slope of the alignment path as before; however, if the accompaniment is not in synch with the performance, we slightly increase/decrease the tempo sequentially in every step. This way, if the alignment starts advancing too fast or too slow due to an inaccurate or extreme curve slope, the method will detect it and start modifying the tempo independently from the slope of the curve, until the accompaniment and the performer are in synch once again.

3.3.3 Implementation details

As mentioned before, all audio playback is handled by a Max/MSP patch, while communication between MATLAB, Max and Java is achieved by messages via the Open Sound Control protocol, or OSC.

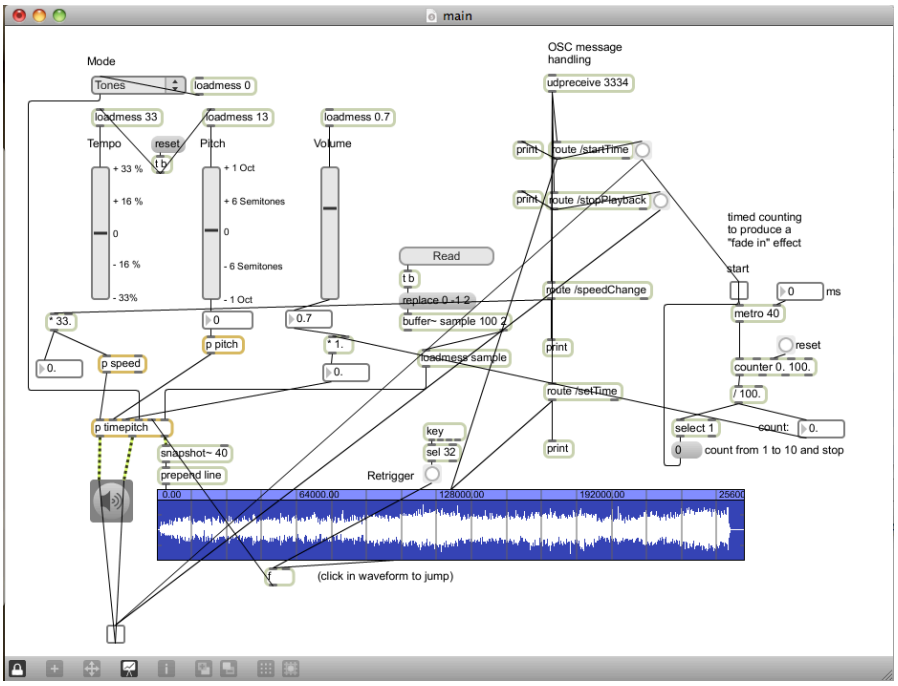

Figure 3. 10 - Screenshot of the Max patch

In order to change the tempo of the accompaniment in real time, we built upon a granular synthesizer patch designed by (Kneppers); unlike spectrum-based time stretching approaches (such as the phase vocoder), granular time stretching has no inherent latency and is therefore suited for real time. A screenshot of the Max/MSP patch can be seen above in Figure 3.10.

4. Results

In this chapter we present some quantitative results for our system. First, results for the lyrics-matching QBH method are presented. Then, we provide a brief discussion regarding the tempo alignment method and its results.

4.1 Lyrics-matching QBH 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.

4.1.1 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 re- trieved phrase and the query can either occur at the query's first or second half, but the duration of the re- trieved 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

$$f_{query} = \frac{l_{query}}{h} \tag{3}$$

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

$$f_{ref} = \frac{l_{ref} - l_{query}}{h} \tag{4}$$

where l_{ref} is the length of the reference in frames. This way, the random guess accuracy can be computed using the following formula:

$$acc = \frac{f_{query}}{f_{ref}} = \frac{l_{query}}{l_{ref} - l_{query}} \tag{5}$$

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

$$acc_2 = \frac{l_{query} * \left[n_r - \left(\frac{n_b}{2}\right) \right]}{l_{ref} - l_{query}} \tag{6}$$

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 should be investigated in the future.

The overall random guess accuracy for each song as well as the average random accuracy can be seen in Table 4.1:

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 random guess accuracy	**0.0471**

Table 4. 1 - Individual and average guess accuracy for the lyrics-matching QBH

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

4.1.2 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 4.2:

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 4. 2 - Accuracy and computed features

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.

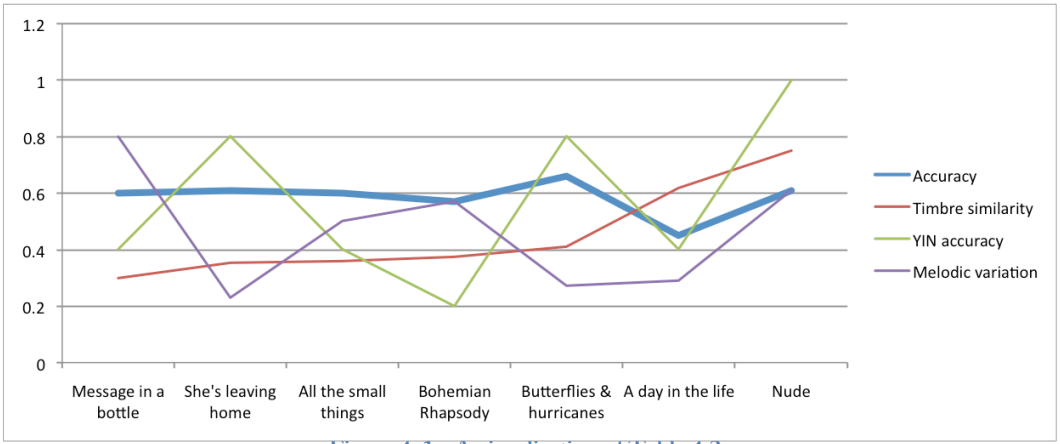

Figure 4. 1 - A visualisation of Table 4.2

It can be seen from Table 4.2, as well as Figure 4.1, that the most apparent 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.

4.2 Discussion on Tempo Alignment results

Quoting from Dixon (Dixon, An on-line time warping algorithm for tracking musical performances, 2005):

"While there is an almost endless supply of professional musical recordings, it is extremely rare to find precise measurements of every played note, which is what we require to assess the accuracy of the alignment. Since manual labeling of a large corpus of music is neither feasible nor accurate, we chose a moderately sized database of piano recordings which were made on a grand piano fitted with infrared sensors which measure precisely the times and velocities of all notes, so that we had audio recordings, discrete measurements of each note and the musical score."

Since our method builds upon the OLTW algorithm, it inherits the same problem; evaluating the alignment would require manually annotating every note in the reference audio as well as the live performance. Moreover, as mentioned before, by the time we reached the point of tempo alignment, most of the time allocated for this project had been devoted to designing and testing the lyrics-matching algorithm.

For this reason, a formal evaluation of the tempo alignment method was not performed; since our algorithm does not interfere with the alignment path estimation

but merely extracts the relative tempo information, we make the assumption that the accuracy of the alignment curve and its slope generated by the OLTW algorithm is analogous to the accuracy of our subsequent tempo alignment. Table 4.3 on the next page presents some results from the evaluation of the OLTW algorithm; however, for more complete results we advise the reader to refer to the evaluation section found in (Dixon & Widmer, MATCH: A music alignment tool chest., 2005).

We did carry out a series of tests in order to qualitatively assess the performance of our method, mainly in order to fine-tune the parameters and test whether or not the method can be applicable to real performance scenarios. These tests consisted of recording deliberate *accelerandos* and *ritardantos* (accelerations and decelerations) of various lengths and intensities, and then running the OLTW algorithm (which produces the same results as with live data).

On this qualitative basis, it can be said that the results produced by our method heavily depend on the complexity of the live performance itself; given that tempo alterations were not abrupt or extreme, the accompaniment successfully follows the user. In cases were the changes in tempo were less smooth, the steep slope produced by the OLTW forward path estimation generated "harsher" tempo adjustments which deteriorated the synchronization between performer and accompaniment, although within time (5-10 seconds, according to the steepness of the path slope) the synchronization was restored.

Composer	Piece (work, section)	Versions	Test Pairs	Events (total)	Error (seconds)		
					Maximum	Mean	Median
Beethoven	Op.15, No.2, bar 1–8	4	6	366	0.70	0.085	0.040
Chopin	Op.15, No.1	13*	77	16863	7.48	0.061	0.020
Mozart	KV279, 1st movt	5	10	5510	5.26	0.036	0.020
Mozart	KV279, 2nd movt	4	6	1836	2.26	0.058	0.020
Mozart	KV279, 3rd movt	5	10	4474	1.38	0.025	0.020
Mozart	KV280, 1st movt	5	10	5990	8.12	0.037	0.020
Mozart	KV280, 2nd movt	5	10	5012	8.90	0.102	0.020
Mozart	KV280, 3rd movt	5	10	2783	4.08	0.044	0.020
Schubert	D899, No.3	12	66	22506	5.74	0.071	0.020
Schumann	Op.15, No.7	6*	14	3570	2.28	0.087	0.020

Table 4. 3 - Dixon and Widmer's quantitative results for the OLTW algorithm

5. Conclusions and future work

In this chapter we provide some thoughts and conclusions on this project, as well as some recommendations for future work based on it.

5.1 Conclusion & discussion

In this work we presented an automatic accompaniment system for the singing voice, addressing the problem of lyrics-matching when querying for melodies. In the core of this system lies the DTW algorithm, which is used as both a similarity measure as well as an alignment tool between the live performance and the reference. The user's position within the piece is initially determined with a variation of current QBH methods, based on three sets of acoustic features; pitch trajectory, RMS energy envelope and MFCCs. After locating the user, the system adapts the tempo of the accompaniment using a novel on-line time warping method.

After a series of adaptations, our system showed a series of interesting results:

- Based on our experiments, the lyrics-matching QBH method produced a Top-1 accuracy of 57.8%, which is promising in the context of a time-critical, single-output system – a similar to our work system by Mesaros et al (Mesaros & Virtanen, 2010) produces a Top-1 accuracy of 57%, although the conditions and scope of their method is rather different than our work.
- The system is feasible for a real-time scenario, as the response time amounts to roughly 70% the duration of the query, with an average query length of 6 seconds. This runtime can be also improved by reducing the sampling rate for the query.
- The tempo alignment, although not quantitatively graded, is fully feasible on real-time scenarios, mainly due to the good performance of the OLTW algorithm.

On the other hand, there are limitations to our work; our evaluation of the lyrics-matching QBH methods has ample room for expansion, mainly in the sense that the system has been tested with a male singer, using male singers as a reference. It would be particularly interesting to see how well our method fares when trying to match a female singer to a male reference, or viceversa.

Finally, there is no quantitative evaluation of the tempo alignment method, mainly due to the fact that our modifications on the OLTW algorithm were very small. However, an indicative evaluation of quantitative nature can be achieved using a small annotated dataset of 1-2 songs, in order to properly determine the success of this method.

5.2 Future work recommendations

The present work has given several interesting and promising results, as presented in the previous chapter. Many of them can be extended and can serve for future research. In this section, we present some of the main points that have to be addressed in the future as refinements or extensions to this work.

5.2.1 A wider-scope lyrics matching system

Seeing as the main contributions of our work lie in the field of QBH, it would be very interesting to see it expand towards a more traditional QBH direction. There are several ways to achieve such a goal:

A first step would be to apply the querying algorithm on a database instead of a single song; this would increase the challenge, and produce interesting results that would help improve the accuracy.

Another direction would be to include source separation techniques, in order to exract vocals from existing songs within a database. This way, this method could be used to jump into any song within a database, at the desired point within the song.

Most improvements and recommendations in the following segment also apply in this case, as they would immediately improve the performance of the lyrics matcher.

5.2.2 From prototype to product

Altough this system can function as a prototype, there are several tasks that would not only increase its performance, but would also possibly prepare it for a commercial application.

An immediate improvement over the lyrics-matching 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, a more thoroughly tested and designed tempo alignment method can be devised, so as to allow the system to cope with more abrupt tempo changes.

6. References

Arzt, A. (2007). Score Following with Dynamic Time Warping; An Automatic Page-Turner. *Master's Thesis, Vienna Institute of Technology.*

Baird, B., Blevins, D., & Zahler, N. (1993). Artificial intelligence and music: Implementing an interactive computer performer. *Computer Music Journal, 17(2):73–79.*

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

Cano, P., Loscos, A., & Bonada, J. (1999). Score-performance matching using HMMs. *In Proceedings of the ICMC.*

Chen, A., Chang, M., & Chen, J. (2000). Query by Music Segments: An Efficient Approach for Song Retrieval. *Proceedings of IEEE International Conference on Multimedia and Expo* .

Dannenberg, R. (1984). An on-line algorithm for real-time accompaniment. *In Proceedings of the International Computer Music Conference, pages 193–198, San Francisco. International Computer Music Association.*

Dannenberg, R., & Hu, N. (2003). Polyphonic audio matching for score following and intelligent audio editors. *In Proceedings of the International Computer Music Conference, pages 27–34, San Francisco. International Computer Music Association.*

de Chevegne, A., & Kawahara, H. (2002). YIN, A fundamental frequency estimator for speech and Music. *The Journal of the Acoustical Society of America, 111:1917.*

Dixon, S. (2005). An on-line time warping algorithm for tracking musical performances. *In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, pages 1727–1728, Edinburgh.*

Dixon, S., & Widmer, G. (2005). MATCH: A music alignment tool chest. *In proceedings of the 6th International Conference on Music Information Retrieval* .

Ellis, D. (2010). Dynamic Time Warping in MATLAB. *http://www.ee.columbia.edu/~dpwe/resources/matla b/dtw/, retrieved on April 2010.*

Foote, J. (1997). 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.*

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

Grubb, L., & Dannenberg, R. (1997). A stochastic method of tracking a vocal performer. *n Proceedings of the International Computer Music Con- ference, pages 301–308, San Francisco. International Computer Music Association.*

Kneppers, M. (n.d.). Max Timestretcher. *http://www.arttech.nl/, Retrieved on June 2010.*

McNab, R. (1996). Towards the Digital Music Library: Tune Retrieval from Acoustic Input. *Proceedings of Digital Libraries, pp 11—18.*

Mesaros, A., & Virtanen, T. (2010). Automatic Recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing vol. 2010, Article ID 546047, 11 pages .*

Orio, N., & Schwarz, D. (2001). Alignment of monophonic and polyphonic music to a score. *In Proceedings of the ICMC, Havana, Cuba.*

Puckette, M. (1995). Explode: A user interface for sequencing and score following. *In Proceedings of the International Computer Music Conference, pages 259– 261, San Francisco. International Computer Music Association.*

Ryynanen, N., & Klapuri, A. (2008). 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.*

Vantomme, J. (1995). Score following by temporal patterns. *Computer Music Journal, 19(3):50–59, 1995.*

Vercoe, B. (1984). The synthetic performer in the context of live performance. *In Proceedings of the International Computer Music Conference, pages 199–200, San Francisco.*

Wang, L., Huang, S., Hu, S., Liang, J., & Xu, B. (2008). 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.*

Wong, C., Szeto, W., & Wong, K. (2007). Automatic lyrics alignment for Cantonese popular Music. *Multimedia Systems, vols. 4-5, no. 12, pp. 307— 323.*