LIVE TRACKING OF MUSICAL PERFORMANCES USING ON-LINE TIME WARPING

Simon Dixon

Austrian Research Institute for Artificial Intelligence
Freyung 6/6, Vienna 1010, Austria
simon@ofai.at

ABSTRACT

Dynamic time warping finds the optimal alignment of two time series, but it is not suitable for on-line applications because it requires complete knowledge of both series before the alignment can be computed. Further, the quadratic time and space requirements are limiting factors even for off-line systems. We present a novel on-line time warping algorithm which has linear time and space costs, and performs incremental alignment of two series as one is received in real time. This algorithm is applied to the alignment of audio signals in order to follow musical performances of arbitrary length. Each frame of audio is represented by a positive spectral difference vector, emphasising note onsets. The system was tested on various test sets, including recordings of 22 pianists playing music by Chopin, where the average alignment error was 59ms (median 20ms). We demonstrate one application of the system: the analysis and visualisation of musical expression in real time.

1. INTRODUCTION

Dynamic time warping (DTW) is a technique for aligning time series which has been well known in the speech recognition community for three decades [1, 2] and has been applied successfully to many other fields ranging from bioinformatics [3] to data mining [4]. DTW has been superseded in speech research by hidden Markov models (HMMs), a related method which uses training data to learn optimal parameter settings.

The DTW algorithm has a time and space complexity which is quadratic in the lengths of the sequences, and this limits its usefulness for matching long sequences. Although simple path constraints can be applied to create a linear time and space algorithm, the resulting solutions are not guaranteed to be optimal with respect to the given cost function. In practice, these solutions are often closer to the desired solution, because they avoid the pathological solutions which can arise as artifacts of imperfect cost functions. However, the use of global path constraints assumes complete knowledge of both sequences, which an on-line algorithm does not have.

Although efficiency and real-time concerns of DTW have been addressed [5], we do not know of any work in which the real-time constraint involves a streamed sequence, so that the alignment must be calculated incrementally, in the forward direction, while one of the sequences is not known in entirety. In section 3 we present our solution to this problem, an on-line time warping algorithm which is able to perform incremental alignment of arbitrarily long sequences in real time.

One challenging application area for on-line time warping is the alignment of audio streams for such tasks as interactive musical performance and visualisation of performance parameters. In section 4, we demonstrate the on-line time warping algorithm in an application for following a musical performance in real time, in which a live performance of a piece of music is aligned to an audio recording of the same piece. This application extracts, analyses and displays parameters reflecting the expression in the performance. Such a system could be used in a concert setting for enhancing the listening experience, or in teaching or rehearsal for providing feedback to the performer.

2. DYNAMIC TIME WARPING

Before presenting our on-line time warping algorithm, we briefly review the standard DTW algorithm, in order to introduce notation and draw attention to some important design choices. A tutorial presentation of DTW can be found in [6]. We present the algorithms in their most general forms in sections 2 and 3, and then in section 4 we apply the on-line algorithm to the specific task of real-time audio alignment.

DTW aligns time series $U = u_1, ..., u_m$ and $V = v_1, ..., v_n$ by finding a minimum cost path $W = W_1, ..., W_l$, where each W_k is an ordered pair (i_k, j_k), such that $(i, j) \in W$ means that 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:

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

Several constraints are placed on W, namely that the path is bounded by the ends of both sequences, and it is monotonic and continuous. Formally:

Bounds:	$W_1 = (1, 1)$
	$W_l = (m, n)$
Monotonicity:	$i_{k+1} \geq i_k$ for all $k \in [1, m-1]$
	$j_{k+1} \geq j_k$ for all $k \in [1, n-1]$
Continuity:	$i_{k+1} \leq i_k + 1$ for all $k \in [1, m-1]$
	$j_{k+1} \leq j_k + 1$ for all $k \in [1, n-1]$

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 [7], 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 [1], which bounds the path with a parallelogram whose long diagonal coincides with the diagonal of the cost matrix. By limiting the slope of the path, either globally or locally, these constraints prevent pathological solutions and reduce the search space.

The minimum cost path can be calculated in quadratic time by dynamic programming, using the recursion:

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

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 $D(m,n)$.

Some formulations introduce various biases in addition to the slope constraints, by multiplying $d(i,j)$ by a weight which is dependent on the direction of the movement. In fact, the above formulation is biased towards diagonal steps: the greater the number of diagonal steps, the shorter the total path length [8, p.177]. We follow Sakoe and Chiba [7] in using a weight of 2 for diagonal steps so that there is no bias for any particular direction:

$$D(i,j) = \min \left\{ \begin{array}{l} D(i,j-1) + d(i,j) \\ D(i-1,j) + d(i,j) \\ D(i-1,j-1) + 2d(i,j) \end{array} \right\} \tag{3}$$

3. ON-LINE TIME WARPING

The quadratic time and space cost is often cited as a limiting factor for the use of DTW with long sequences, but the widely used global path constraints can be trivially modified to create a linear time and space algorithm. For instance, if the width of the Sakoe-Chiba bound is set to a constant rather than a fraction of the total length, the number of calculations becomes linear in the length of the sequences. The danger with this approach is that it is not known how close to the diagonal the optimal solution is, so the desired solution might be excluded by a band around the diagonal which is too narrow.

However, in the case of on-line alignment of a known time series with a partially unknown series, several changes must be made to the DTW algorithm. 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. Another change is that we require an incremental solution, so the minimum cost path must be calculated in the forward direction. Further, 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.

Using the notation from section 2, suppose U is the partially unknown sequence. Then at each time t (measured in integer units corresponding to the indices of U), we seek the best alignment of the sequence $u_1, ..., u_t$ to some initial subsequence of V. Our solution fulfilling the above conditions is the on-line time warping algorithm given in Figure 1. It has one parameter, c, which determines the width of the search band. The variables t and j are pointers to the current positions in series U and V respectively, which are initialised to point to the start of each series.

The main loop of the algorithm calculates a partial row or column of the path cost matrix. The calculation of the path cost uses the standard DTW recursion formula, restricted to use only the matrix entries which have already been calculated. The path cost is normalised by the path length, so that paths of varying lengths can be compared in the function $GetInc$. The number of cells calculated is given by the parameter c. If a new row is being calculated, the row number is incremented, and the cells in the last c columns up to and including the current column are calculated. Figure 2 illustrates a sequence of row and column calculations.

The function $GetInc$ selects whether to calculate a row, column, or both. If less than c elements of each series have been processed, new rows and columns are alternately calculated (Figure 2, steps 1–7). If one sequence has been incremented successively $MaxRunCount$ times, the other sequence is incremented

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 1: *The on-line time warping algorithm. See text for explanations.*

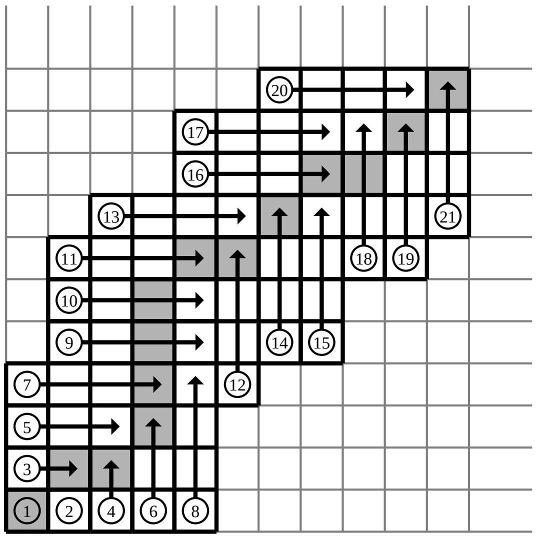

Figure 2: *An example of the on-line time warping algorithm with search window $c = 4$, showing the order of evaluation for a particular sequence of row and column increments. The axes represent the variables t and j (see Figure 1) respectively. All calculated cells are framed in bold, and the optimal path is coloured grey.*

(step 12). Otherwise the minimum path cost for each cell in the current row and column is found. If this occurs in the current position (t, j), then both the row and column counts are incremented (e.g., steps 20–21); if it occurs elsewhere in row j, then the row count is incremented (e.g., step 10), otherwise the column count t is incremented (e.g., step 19). This enables dynamic tracking of the minimum cost path using a small fixed width band around a varying "diagonal".

Since the on-line time warping algorithm cannot look into the future, its alignment path must be calculated in the forward direction. In the algorithm above, the function $GetInc$ calculates the current optimal path as ending at the point (x, y), which we call the current alignment point. Now, if the kth alignment point is (x_k, y_k), there is no way of knowing if this point will lie on the optimal path for $k' > k$. Further, there is no guarantee of continuity between the paths of length $k - 1$ and k, nor in the sequence of alignment points $(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)$.

Two approaches can be taken to address this problem. First, if the application allows a certain amount of latency, then the choice of alignment points can be based on a limited view into the future. That is, for path length $k + \delta$, we output the point (x'_k, y'_k), the kth point on the optimal path to $(x_{k+\delta}, y_{k+\delta})$, which might be different to the point (x_k, y_k) calculated for path length k. For increasing values of δ, the path becomes increasingly smooth and closer to the global optimum computed by the reverse path algorithm of DTW. The second approach applies smoothing directly to the sequence of alignment points. This requires no future information, but it still builds an effective latency into the system. (If the smoothing function is interpreted as a filter, the latency is equal to its group delay.) In the system described in section 4, neither approach was deemed necessary, since if the forward path estimation is correct, no retrospective adjustment of the path is necessary, and the path consisting of the current alignment points is continuous.

3.1. Efficiency and Correctness

For each new row or column, the on-line time warping algorithm calculates up to c cells and makes less than $2c + MaxRunCount$ comparisons. We are specifically interested in the behaviour with respect to the arrival of a new element u_t. As long as the slope of the sequence of increments is bounded (i.e. by $MaxRunCount$), then the number of calculations to be performed for each time t is bounded by a constant.

The correctness of the algorithm (in terms of finding the globally minimal path) cannot be guaranteed without calculating the complete distance matrix. Thus, any path constraint immediately denies this sense of optimality, but as stated previously, minimum cost paths with large singularities are usually undesired artifacts of an imperfect cost function. For each incoming data point u_t, the minimum cost path calculated at time t is the same as that calculated by DTW, assuming the same path constraints. The advantage of the on-line algorithm is that the centre of the search band is adaptively adjusted to follow the best match, which allows a smaller search band than the standard bands around a fixed diagonal.

4. TRACKING OF MUSICAL EXPRESSION

In music performance, high level information such as structure and emotion is communicated by the performer through a range of different parameters, such as tempo, dynamics, articulation and vibrato. These parameters vary within a musical piece, between musical pieces and between performers. An important step towards modelling of this phenomenon is the measurement of the expression parameters in human musical performance, which is a far from trivial task [9, 10]. Since we do not anticipate that the great musicians would perform with sensors attached to their fingers or instruments (!), we wish to extract this information directly from the audio signal.

State of the art audio analysis algorithms are unable to reliably extract precise performance information, so a hand-correction step is often employed to complement the automatic steps. This step is labour intensive, error-prone, and only suitable for off-line processing, so we propose using automatic alignment of different performances of the same piece of music as a key step in extracting performance parameters. In the off-line case, automatic alignment enables comparative studies of musical interpretation directly from audio recordings [11]. If one performance is already matched to the score, it can then be used as a reference piece for the extraction of absolute measurements from other performances. Further, one could synthesise a performance directly from the score, and avoid the initial manual matching of score and performance entirely. Of particular interest in this paper is the on-line case, the live tracking and visualisation of expressive parameters during a performance. This could be used to complement the listening experience of concert-goers, to provide feedback to teachers and students, and to implement interactive performance and automatic accompaniment systems. In this section we describe the implementation of a real-time performance alignment system using on-line time warping, concluding with an example of an application for tracking and visualising musical expression.

4.1. Cost Function

The alignment of audio files is based on a cost function which assesses the similarity of frames of audio data. We use a low level spectral representation of the audio data, generated from a windowed FFT of the signal. A Hamming window with a default size of 46ms (2048 points) is used, with a hop size of 20ms. The spectral representation was chosen over a higher level symbolic representation of the music in order to avoid a pitch recognition step, which is notoriously unreliable in the case of polyphonic music. The frequency axis was mapped to a scale which is linear at low frequencies and logarithmic at high frequencies. This achieved a significant data reduction without loss of useful information, at the same time mimicking the linear-log frequency sensitivity of the human auditory system. The lowest 34 FFT bins (up to 370Hz, or F♯4) were mapped linearly to the first 34 elements of the new scale. 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 (G9) were summed into the last element of the new scale. The resulting vector contained a total of 84 points instead of the original 2048.

The most important factor for alignment is the timing of the onsets of tones. The subsequent evolution of the tone gives little information about the timing and is difficult to align using energy features, which change relatively slowly over time within a note. Therefore 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, and these positive spectral difference vectors are compared using the Euclidean distance:

$$d(i,j) = \sqrt{\sum_{b=1}^{84} (E'_u(b,i) - E'_v(b,j))^2} \tag{4}$$

where $E'_x(f,t)$ represents the increase in energy $E_x(f,t)$ of the signal $x(t)$ in frequency bin f at time frame t:

$$E'_x(f,t) = \max(E_x(f,t) - E_x(f,t-1), 0) \tag{5}$$

4.2. Implementation Details

For performance alignment, the slope of the path represents the relative tempo of the two pieces. Since musical performances are not arbitrarily fast, it is reasonable to constrain the slope to a range between $\frac{1}{3}$ and 3 ($MaxRunCount = 3$), which allows for large but not arbitrary differences in tempo. The on-line time warping algorithm uses a search width of 10 seconds (c = 500 frames). This is much larger than any error we have encountered in testing, but it still runs comfortably in real time on a 3GHz PC. The software is implemented in Java, and in off-line tests with the above parameters it aligned each minute of music in about 5 seconds. The software can be downloaded from:

`http://www.ofai.at/~simon.dixon/match`

4.3. Quantitative Evaluation

It is not possible to perform quantitative testing on a live performance without a reference alignment, so we present quantitative results from off-line tests; the results would be the same with live data. 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 labelling 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 (Bösendorfer SE290) 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. The recordings consist of 22 pianists playing 2 excerpts of solo piano music by Chopin (Etude in E Major, Op.10, no.3, bars 1–21; and Ballade Op.38, bars 1–45) [12]. The pianists were students or professors at the Vienna Music University. The Etude performances ranged from 70.1 to 94.4 seconds duration, and the Ballade ranged from 112.2 to 151.5 seconds. Such significant differences in tempo would not be compatible with the use of global path constraints.

Error $\leq$		Cumulative error counts			
Frames	Seconds	On-line		Off-line	
		Notes	Percent	Notes	Percent
0	0.00	18774	22.4%	38655	46.1%
1	0.02	45822	54.7%	72934	87.1%
2	0.04	61019	72.8%	79126	94.5%
3	0.06	68380	81.6%	80540	96.2%
5	0.10	74325	88.7%	81343	97.1%
10	0.20	79001	94.3%	82325	98.3%
25	0.50	82486	98.5%	83292	99.4%
50	1.00	83581	99.8%	83658	99.9%

Table 1: Results for pairwise alignment of 22 performances of 2 Chopin piano pieces, shown as cumulative counts and percentages of notes with an error up to the given value (see text).

The results were calculated as follows. For each chord (set of simultaneous notes according to the musical notation), the average onset time was calculated, and the corresponding chords in the two performances were aligned using the symbolic data. This gave a set of points through which the time warping path should pass. The error for each point was calculated as the Manhattan distance (sum of horizontal and vertical displacements) of the nearest point on the time warping path. In Table 1, we show the error counts for errors less than or equal to 0,1,2,3,5,10,25 and 50 frames across the 462 ($= 2 \times \frac{22 \times 21}{2}$) pairs of performances. For comparison, the results using the off-line version of the algorithm are included in the table [11]. The average error was 59ms for the on-line algorithm and 23ms for the off-line version; the worst errors were 3.16s (on-line) and 2.82s (off-line). The off-line algorithm performs better because it has the advantage of knowing the future evolution of the signal when calculating the alignment at a given point. These results are comparable with human ability: the human temporal order threshold (the ability to distinguish the order of two sounds occurring closely in time) is approximately 40ms, and can be much worse in the context of annotating musical recordings [13]. Further test results for the off-line system are presented in [11].

4.4. Live Visualisation of Musical Expression

One application of the alignment algorithm is to provide live analysis of expression in musical performance. At the conference we can demonstrate the system tracking tempo and dynamics in a live performance and displaying the data with an animation designed by musicologists for off-line performance visualisation [14], as shown in Figure 3. This system could be used in a concert setting, for enhancing the audience's understanding of the musician's interpretation of the music, in comparative studies of interpretation, or in teaching or private rehearsal.

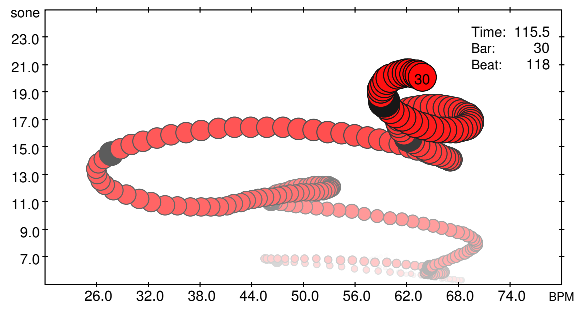

Figure 3: *The Performance Worm is an animation of some of the expressive parameters of a musical performance. The position of the head of the worm gives the current tempo (horizontal axis) and loudness level (vertical axis). The figure shows the display at bar 30 of Rachmaninov's Prelude op.23 no.6 played by Vladimir Ashkenazy.*

5. DISCUSSION AND CONCLUSION

We presented a new on-line time warping algorithm, which aligns a sequence arriving in real time with a stored sequence of arbitrary length. DTW finds a path through a cost matrix which is optimal with respect to the sum of the match costs along the path. We modified the DTW algorithm to calculate the optimal path incrementally in real time as one of the sequences is received. We are not aware of any other time warping algorithm which has these properties; some similar ideas are found in the overlap matching algorithm outlined in [3, pp. 26–27] and the longest common subsequence dynamic programming method used in [15]. At each time frame, the calculated path is optimal with respect to the data computed up to that time, but this might not correspond to the optimal path calculated by an off-line algorithm with full knowledge of both sequences.

The on-line time warping algorithm was used in implementing a musical performance alignment system. Tests with several hundred pairs of performances of two piano pieces confirmed that the method is suitable for this type of real-time application.

The performance alignment system uses a low-level representation for music. An alternative would be to use a pitch extraction algorithm to enable matching to be performed on the level of musical symbols. However, pitch extraction for polyphonic music (and particularly for multiple instruments) is still too unreliable in this context, and large errors would occur when notes were missed. Also, the main advantage of a high-level representation is that it affords a data reduction of several orders of magnitude, but we have shown that it is possible to write an efficient time warping system based on a low level representation.

The cost function was based on derivative spectral features, in order to emphasise tone onsets. Derivative features have been used in speech recognition [7], are advocated generally by Keogh and Pazzani [16], and they have been used in score following [17]. Other authors propose the use of a chromogram [18], which reduces the frequency scale to twelve pitch classes, independent of octave. This might be suitable for retrieval by similarity, where absolute identity of matching musical pieces is not assumed, and a large number of comparisons must be performed in a short time, but it discards much more information than is necessary for real-time alignment. Whether this results in a loss of accuracy remains to be shown. Other features such as Mel frequency cepstral coefficients (MFCCs) are often used in speech and audio research, but they capture the spectral shape (reflecting the timbre of the instrument) rather than the pitch (reflecting the notes that were played).

In the music field, DTW has been used for score-performance alignment [17, 19, 20] and query by humming applications [21, 22] which do not use on-line algorithms. The earliest score following systems used dynamic programming [15], based on a high-level symbolic representation of the performance which was only usable with monophonic audio. Alternative approaches use hidden Markov models [23, 24] and hybrid graphical models [25], which both require training data for each piece. The test data used in this paper is somewhat exceptional; in general, we will not have access to multiple labelled performances.

The test results gave an average accuracy of 59ms (median 20ms), which is sufficient for most purposes. The largest errors occurred at the beginnings of the pieces, where there is not yet sufficient data for alignment, and at phrase boundaries, where there are large variations in tempo (i.e., discontinuities in timing). The forward path calculation produces a path that is less smooth than that produced by DTW. However, experiments with smoothing (e.g., using a least squares fit) did not increase the accuracy at the note level, since most of the unsmoothness occurs between note onsets, and thus is irrelevant to the assessment of accuracy, which only considers note onset times.

The test recordings were all made under consistent recording conditions (same piano, microphone and room), making alignment easier. Further tests have been performed recently, using other pieces, recording conditions and instruments (including orchestral and popular music). The alignment was successful in over 99% of test cases, but the errors were higher (median error 80ms for the on-line algorithm, compared with 20ms for the off-line version) [11].

One limitation of the on-line alignment algorithm is that it does not allow for structural differences, such as insertions, deletions or repetitions of sections or phrases. These changes would go beyond the boundaries of the search window in the current implementation. A possible solution would be to implement a method for dealing with alternate paths, such as directed networks [8, ch. 10], but this would not function within the current real-time constraints without limiting the lengths of pieces and reducing the timing resolution.

The alignment system can be used off-line for relative comparisons of performance interpretation, using unlabelled audio recordings. One application is a media player plug-in which, given a position in one audio file, automatically finds the corresponding position in other audio files of the same piece of music [11]. Such an innovation would be welcomed by music performance researchers and classical music lovers.

Another application is an automatic accompaniment system. In the case where no score is available but an audio file exists with the soloist and accompaniment on separate tracks (which would be easy for record companies to produce), alignment could be performed on the solo track while the accompaniment track could be played back with corresponding accommodation of the dynamic and timing changes of the soloist. One possible scenario is a virtual karaoke system, such as in [26]. Currently planned extensions to this work include a score following system and a range of visualisation tools for use in concerts, teaching and private rehearsal.

6. ACKNOWLEDGEMENTS

This work was supported by the Vienna Science and Technology Fund as part of the project CI-010 *Interfaces to Music*, the European Union project EU-FP6-IST-507142 SIMAC, and the START project Y99-INF supported by the Austrian Federal Ministry of Education, Science and Culture (BMBWK). The Austrian Research Institute for Artificial Intelligence acknowledges the financial support of the BMBWK and the Austrian Federal Ministry of Transport, Innovation and Technology.

7. REFERENCES

[1] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 23, pp. 52–72, 1975.

[2] L. R. Rabiner, A. Rosenberg, and S. Levinson, "Considerations in dynamic time warping algorithms for discrete word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, pp. 575–582, 1978.

[3] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Neucleic Acids*, Cambridge University Press, Cambridge UK, 1998.

[4] D. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *AAAI Workshop on Knowledge Discovery in Databases*, 1994, pp. 229–248.

[5] S. Salvador and P. Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," in *KDD Workshop on Mining Temporal and Sequential Data*, 2004.

[6] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice, Englewood Cliffs, NJ, 1993.

[7] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimisation for spoken word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, pp. 43–49, 1978.

[8] D. Sankoff and J.B. Kruskal, *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*, Addison-Wesley, New York/Menlo Park/Reading, 1983.

[9] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[10] S. Dixon, "On the analysis of musical expression in audio signals," in *Storage and Retrieval for Media Databases 2003, SPIE-IS&T Electronic Imaging Conference*, M.M. Yeung, R.W. Leinhart, and C-S. Li, Eds., 2003, vol. SPIE Volume 5021, pp. 122–132.

[11] S. Dixon and G. Widmer, "MATCH: A music alignment tool chest," in *6th International Conference on Music Information Retrieval*, 2005, to appear.

[12] W. Goebl, "Melody lead in piano performance: Expressive device or artifact?," *Journal of the Acoustical Society of America*, vol. 110, no. 1, pp. 563–572, 2001.

[13] S. Dixon, W. Goebl, and E. Cambouropoulos, "Smoothed tempo perception of expressively performed music," *Music Perception*, vol. 23, 2005, to appear.

[14] J. Langner and W. Goebl, "Visualizing expressive performance in tempo-loudness space," *Computer Music Journal*, vol. 27, no. 4, pp. 69–83, 2003.

[15] R.B. Dannenberg, "An on-line algorithm for real-time accompaniment," in *Proceedings of the International Computer Music Conference*, 1984, pp. 193–198.

[16] E.J. Keogh and M.J. Pazzani, "Derivative dynamic time warping," in *SIAM International Conference on Data Mining*, 2001.

[17] N. Orio and D. Schwarz, "Alignment of monophonic and polyphonic music to a score," in *Proceedings of the International Computer Music Conference*, 2001, pp. 155–158.

[18] R.B. Dannenberg and N. Hu, "Polyphonic audio matching for score following and intelligent audio editors," in *Proceedings of the International Computer Music Conference*, 2003, pp. 27–34.

[19] F. Soulez, X. Rodet, and D. Schwarz, "Improving polyphonic and poly-instrumental music to score alignment," in *4th International Conference on Music Information Retrieval*, 2003, pp. 143–148.

[20] R. Turetsky and D. Ellis, "Ground-truth transcriptions of real music from force-aligned MIDI syntheses," in *4th International Conference on Music Information Retrieval*, 2003, pp. 135–141.

[21] D. Mazzoni and R.B. Dannenberg, "Melody matching directly from audio," in *2nd International Symposium on Music Information Retrieval*, 2001, pp. 73–82.

[22] Y. Zhu and D. Shasha, "Warping indexes with envelope transforms for query by humming," in *ACM SIGMOD Conference*, 2003.

[23] P. Cano, A. Loscos, and J. Bonada, "Score-performance matching using HMMs," in *Proceedings of the International Computer Music Conference*. 1999, pp. 441–444, International Computer Music Association.

[24] N. Orio and F. Déchelle, "Score following using spectral analysis and hidden Markov models," in *Proceedings of the International Computer Music Conference*, 2001, pp. 151–154.

[25] C. Raphael, "A hybrid graphical model for aligning polyphonic audio with musical scores," in *Proceedings of the 5th International Conference on Musical Information Retrieval*, 2004, pp. 387–394.

[26] P. Cano, A. Loscos, J. Bonada, M. de Boer, and X. Serra, "Voice morphing system for impersonating in karaoke applications," in *Proceedings of the International Computer Music Conference*. 2000, International Computer Music Association.