An On-Line Time Warping Algorithm for Tracking Musical Performances[*]

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

Abstract

Dynamic time warping is not suitable for on-line applications because it requires complete knowledge of both series before the alignment of the first elements can be computed. 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 track musical performances.

1 On-Line Time Warping

Although efficiency and real-time concerns of dynamic time warping (DTW) have been addressed in the literature, 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 this work we present an on-line time warping (OLTW) algorithm which is able to perform incremental alignment of arbitrarily long sequences in real time.

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 $d(i, j)$, usually represented as an $m \times n$ matrix, which assigns a match cost for aligning each pair (u_i, v_j). The path cost is the sum of the local match costs along the path. Several constraints are placed on W, namely that the path is bounded by the ends of both sequences, and it is monotonic and continuous. The minimum cost path can be calculated in quadratic time by dynamic programming, using the recursion:

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

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 $D(m, n)$.

In the on-line case, the length of the incoming sequence is unknown, so one of the boundary conditions for the search must be estimated along with the optimal path. Also, 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. If U is the partially unknown sequence, then we seek at each time t the best alignment of the sequence $u_1, ..., u_t$ to some initial subsequence of V. This is performed with the OLTW algorithm (Figure 1), which we now explain.

The variables t and j are pointers to the current positions in series U and V respectively. 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 search width parameter, c; e.g. for a new row, the row number is incremented, and the cells in the last c columns up to and including the current column are calculated.

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. If one sequence has been incremented successively $MaxRunCount$ times, the other sequence is incremented. 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; if it occurs elsewhere in row j, then the row count is incremented, otherwise the column count t is incremented.

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, but the number of calculations performed by OLTW is bounded by a constant. A further advantage of OLTW is that the centre of the search band is adaptively adjusted to follow the best match.

2 Tracking of Musical Performances

In music performance, high level information such as structure and emotion is communicated by the performer through parameters such as tempo, dynamics, articulation and vibrato. These parameters vary within a musical piece, between musical pieces and between performers. We use OLTW to extract this information directly from audio signals by aligning different performances of the same piece of music, enabling live tracking and visualisation of expressive parameters during a performance. This could be used to complement the listening

[*]This work was supported by the Vienna Science and Technology Fund project CI010 *Interfaces to Music* and the EU-FP6-IST-507142 project *SIMAC*. ÖFAI acknowledges the financial support of the Austrian federal ministries BMBWK and BMVIT.

```
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 (OLTW) algorithm.

experience of concert-goers, to provide feedback to teachers and students, or to enable interactive performance and automatic accompaniment systems.

The audio data is represented by the positive spectral difference between successive 20ms frames, that is, the increase in energy (if any) in each frequency bin. This emphasises the onsets of tones, the most important indicators of musical timing. The cost of matching two frames is given by the Euclidean distance between their spectral difference vectors.

Since the slope of the path represents the relative tempo, it is reasonable to constrain it to a range between $\frac{1}{3}$ and 3 $(MaxRunCount = 3)$, to allow for moderate but not arbitrary differences in tempo. In off-line tests with a search width of 10s (c = 500 frames) the program aligned each minute of music in about 10 seconds (3GHz PC). It also runs comfortably in real time with these parameters.

Error $\leq$		Percentage of notes			
Frames	Seconds	On-line		Off-line	
		Etude	Ballade	Etude	Ballade
0	0.00	13.9%	13.1%	46.5%	36.6%
1	0.02	36.1%	34.5%	84.5%	77.1%
2	0.04	52.8%	50.7%	91.1%	88.9%
3	0.06	64.0%	62.2%	93.4%	92.5%
5	0.10	78.4%	76.7%	96.0%	95.1%
10	0.20	92.7%	91.4%	98.8%	97.4%
25	0.50	98.7%	98.5%	99.8%	99.1%
50	1.00	99.9%	99.7%	100.0%	99.8%

Table 1: Alignment results shown as cumulative percentages of notes with an error up to the given value (see text).

Quantitative testing was performed off-line with recordings 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), played on a computer-monitored grand piano. This provided precise measurements of the times and velocities of all notes, so that we had both the audio recordings and discrete measurements of each note.

After aligning a pair of files, the error for each note or chord was calculated, using the Manhattan distance between the point representing the onsets of the corresponding notes in the two performances and the nearest point on the time warping path. In Table 1, we show the percentages of notes with errors less than or equal to 0,1,2,3,5,10,25 and 50 frames across the 231 ($= \frac{22\times21}{2}$) pairs of performances of each piece. The results using the off-line version of the algorithm are also included in the table. The average error was 84ms for the on-line algorithm and 30ms for the off-line version; the worst errors were 2.62s (on-line) and 3.64s (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.

3 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. 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.

OLTW was used in implementing a musical performance alignment system, which was tested on several hundred pairs of performances with an average error of 84ms. The system can be used off-line for comparisons of performance interpretation, using unlabelled audio recordings, e.g. in 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. An on-line application is automatic accompaniment: given an audio file with the soloist and accompaniment on separate tracks, alignment could be performed on the solo track while the accompaniment track is played back with corresponding accommodation of the dynamic and timing changes of the soloist. Currently planned extensions of this work include a score following system and a range of visualisation tools for use in concerts, teaching and private rehearsal. At the conference we will 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.