ACCURATE REAL-TIME WINDOWED TIME WARPING

Robert Macrae
Centre for Digital Music
Queen Mary University of London
robert.macrae@elec.qmul.ac.uk

Simon Dixon
Centre for Digital Music
Queen Mary University of London
simon.dixon@elec.qmul.ac.uk

ABSTRACT

Dynamic Time Warping (DTW) is used to find alignments between two related streams of information and can be used to link data, recognise patterns or find similarities. Typically, DTW requires the complete series of both input streams in advance and has quadratic time and space requirements. As such DTW is unsuitable for real-time applications and is inefficient for aligning long sequences. We present Windowed Time Warping (WTW), a variation on DTW that, by dividing the path into a series of DTW windows and making use of path cost estimation, achieves alignments with an accuracy and efficiency superior to other leading modifications and with the capability of synchronising in real-time. We demonstrate this method in a score following application. Evaluation of the WTW score following system found 97.0% of audio note onsets were correctly aligned within 2000 ms of the known time. Results also show reductions in execution times over state-of-the-art efficient DTW modifications.

1. INTRODUCTION

Dynamic Time Warping (DTW) is used to synchronise two related streams of information by finding the lowest cost path linking feature sequences of the two streams together. It has been used for audio synchronisation [3], cover song identification [13], automatic transcription [14], speech processing [10], gesture recognition [7], face recognition [1], lip-reading [8], data-mining [5], medicine [15], analytical chemistry [2], and genetics [6], as well as other areas. In DTW, dynamic programming is used to find the minimal cost path through an accumulated cost matrix of the elements of two sequences. As each element from one sequence has to be compared with each element from the other, the calculation of the matrix scales inefficiently with longer sequences. This, combined with the requirement of knowing the start and end points of the sequences, makes DTW unsuitable for real-time synchronisation. A real-time variant would make DTW viable at larger scales and capable of driving applications such as score following, automatic accompaniment and live gesture recognition.

Local constraints such as those by Sakoe and Chiba [10] improve the efficiency of DTW to linear time and space complexity by limiting the potential area of the accumulated cost matrix to within a set distance of the diagonal. However, not all alignments necessarily fit within these bounds. Salvador and Chan proposed, in FastDTW [11], a multi-resolution DTW where increasingly higher resolution DTW paths are bounded by a band around the previous lower resolution path, leading to large reductions in the execution time. On-Line Time Warping by Dixon [3] made real-time synchronisation with DTW possible by calculating the accumulated cost in a forward manner and bounding the path by a forward path estimation.

While the efficiency of DTW has been addressed in FastDTW [11] and the real-time aspect has been made possible with On-Line Time Warping [3], WTW contributes to synchronisation by offering steps to further improve the efficiency whilst working in a progressive (real-time applicable) manner and preserving the accuracy of standard DTW. This method consists of breaking down the alignment into a series of separate bounded sub-paths and using a cost estimation to limit the area of the accumulated cost matrix calculated to small regions covering the alignment.

In Section 2 we explain conventional DTW before describing how WTW works in Section 3. In Section 4 we evaluate the accuracy and efficiency of WTW in a score following application. Finally, in Section 5, we draw conclusions from this work and discuss future improvements.

2. DYNAMIC TIME WARPING

DTW requires two sets of features to be extracted from the two input pieces being aligned and a function for calculating the similarity between any two frames of these feature sets. One such measurement of the similarity is the inner product. As the inner product returns a high value for similar frames, we subtract the inner product from one so that the optimal path cost is the path with the minimal cost. Equation 1 shows how to calculate this similarity measurement between frames A_m and B_n from feature sequences $A = (a_1, a_2, ..., a_M)$ and $B = (b_1, b_2, ..., b_N)$ respectively:

$$d_{A,B}(m, n) = 1 - \frac{< a_m, b_n >}{\|a_m\|\|b_n\|} \qquad (1)$$

Dynamic programming is used to find the optimum path, $P = (p_1, p_2, ..., p_W)$, through the similarity matrix $\mathcal{C}(m, n)$

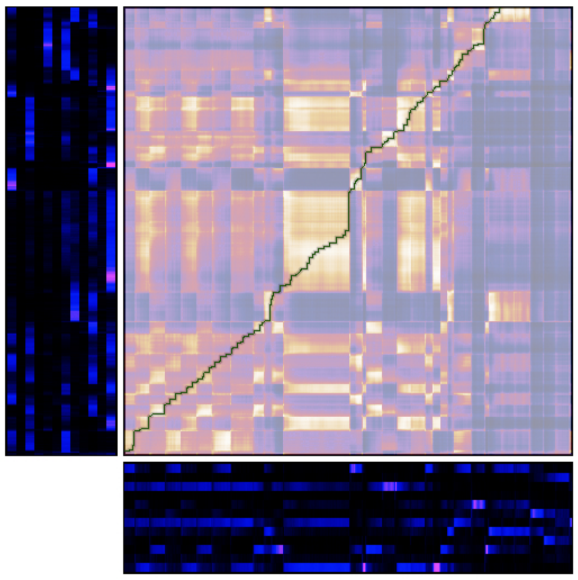

Figure 1. Dynamic Time Warping aligning audio with a musical score. The audio is divided into chroma frames (bottom) which are then compared against the score's chroma frames (left). The similarity matrix (centre) shows a path where the sequences have the lowest cost (highest similarity). Any point on this path indicates where in the score the corresponding audio relates to.

with $m \in [1 : M]$ and $n \in [1 : N]$ where each $p_k = (m_k, n_k)$ indicates that frames a_{m_k} and b_{n_k} are part of the aligned path at position k. An example of this similarity matrix, including the features used and the lowest cost path, can be seen in Figure 1. The final path is guaranteed to have the minimal overall cost $D(P) = \sum_{k=1}^{W} d_{A,B}(m_k, n_k)$, within the limits of the features used, whilst satisfying the following conditions:

Bounds:	$p_1 = (1, 1)$
	$p_W = (M, N)$
Monotonicity:	$m_{k+1} \geq m_k$ for all $k \in [1, W-1]$
	$n_{k+1} \geq n_k$ for all $k \in [1, W-1]$
Continuity:	$m_{k+1} \leq m_k + 1$ for all $k \in [1, W-1]$
	$n_{k+1} \leq n_k + 1$ for all $k \in [1, W-1]$

3. WINDOWED TIME WARPING

WTW consists of calculating small sub-alignments and combining these to form an overall path. Subsequent sub-paths are started from points along the previous sub-paths. Real-time path positions can then be extrapolated from these sub-paths. The end points of these sub-alignments are either undirected, by assuming they lie on the diagonal, or directed, by using a forward path estimate. As such WTW can be seen as a two-pass system similar to FastDTW and OTW. The sub-alignments make use of an optimisation that avoids calculating points with costs that are over the cost estimate (provided by the initial direction path), referred to as the A-Star Cost Matrix. WTW also requires the use of Features, Window Dimensions, and Local Constraints that all affect how the alignments are made. The overall process is outlined in Algorithm 1. In order to demonstrate WTW we implemented a score following application using this method to synchronise audio and musical scores.

```
Input: Feature Sequence A and Feature Sequence B
Output: Alignment Path
Path = new Path.starting(1,1);
while Path.length < min (A.length,B.length) do
    Start = Path.end;
    End = Start;
    while (End - Start).length < Window_Size do
        End =
        argmin(Inner_Product(End.next_points));
    end
    Cost_Estimate = End.cost;
    A-Star_Matrix =
    A_Star_Fill_Rect(Start,End,Cost_Estimate);
    Path.add(A_Star_Matrix.getPath(1,Hop_Size));
end
return Path;
```

Algorithm 1: The Windowed Time Warping algorithm.

3.1 Features

The feature vector describes how the sequence data is represented and segmented. The sequence is divided up into feature frames in order to differentiate the changes in the sequence over time. The frame size and spacing are referred to as the window size and hop size respectively. The implementation of WTW for score following requires a musically based feature vector. In this case, we use chroma features, a 12 dimensional vector corresponding to the unique pitch classes in standard Western music. The intensities of the chroma vectors can be seen as a representation of the harmonic and melodic content of the music. In our implementation we use a window size of 200ms and a hop size of 50ms.

3.2 Window Dimensions

Similar to how the sequence data is segmented, the windows of standard DTW in WTW have a window size and hop size to describe their size and spacing respectively. A larger window size and/or smaller hop size will increase the accuracy of the alignment, as more of the cost matrix is calculated, however will this will be less efficient. Examples of different window and hop sizes can be seen in Figure 2 and a comparison of Window and Hop sizes is made in Section 4.

3.3 Local Constraints

We refer to two types of local constraints in Dynamic Programming. The first, henceforth known as the *cost constraint*, indicates the possible predecessors of a point p_k on a path. The predecessor p_{k-1} with lowest path cost $D(p_{k-1})$ is chosen when calculating the accumulated cost

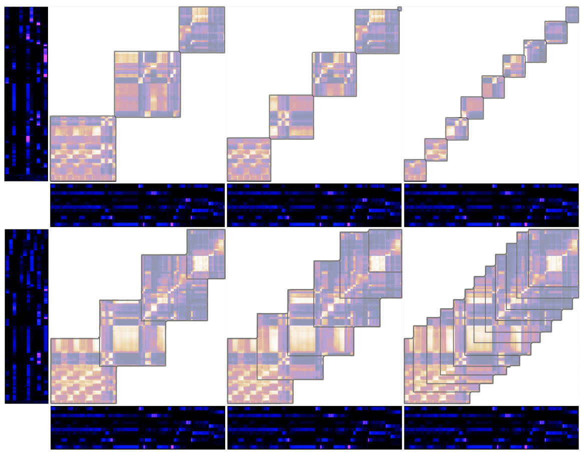

Figure 2. The regions of the similarity matrix computed for various values of the window size (top row) and hop size (bottom row).

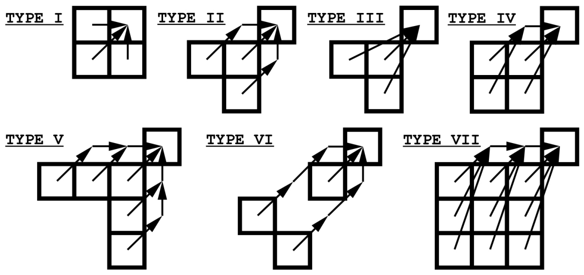

Figure 3. Some example local constraints as defined by Rabiner and Juang [9].

matrix. The second, referred to as the *movement constraint*, indicates the possible successors of a point p_k. Standard DTW doesn't make use of a *movement constraint* as all the frames in the cost matrix are calculated. Examples of local constraints by Rabiner and Juang [9] are show in Figure 3. These constraints define the characteristics of the dynamic programming. For example, Type I allows for horizontal and vertical movement which corresponds to a single frame of one sequence being linked to multiple frames of the other. All the other Types allow high cost frames to be skipped and Type III and II show how the paths can skip these frames directly or add in the single steps, respectively. The two path finding algorithms, described next, make use of the Type I and a modified version of the Type VII (where the steps are taken directly as in Type III) local constraints.

3.4 Window Guidance

The sequential windows that make up the alignment of WTW can be either directed or undirected. Whilst it can help to direct the end point of the windows of DTW (particularly for alignments between disproportional sequences where the expected path angle will be far from 45°), the sub-paths calculated within these windows can make up for an error in the estimation. A low hop size should ensure the point taken from the sub-path as the starting point for the next window is likely to be on the correct path.

For the windows to be directed, a forward estimation is required. The Forward Greedy Path (FGP) is an algorithm which makes steps through the similarity matrix based on whichever subsequent step has the highest similarity (minimal cost) using a *movement constraint* to decide which frames are considered. In this manner the path can work in an efficient forward progressive manner, however, will be more likely to be thrown off the correct path by any periods of dissimilarity within the alignment. The first FGP path $F = (f_1, f_2, ..., f_W)$ where $f_k = (m_k, n_k)$ starts from position $f_1 = (m_1, n_1)$ and from then on each subsequent frame is determined by whichever of the available frames, as determined by the local constraint, has the lowest cost. Therefore the total cost $D(m, n)$ to any point (m, n) on the FGP path F is $D(f_k) = \sum_{l=1}^{k} d(f_l)$ and any point is dependent on the previous point: $f_{k+1} = argmin(d(i, j))$ where the range of possible values for i and j are determined by f_k and the local constraints.

The FGP path only needs to calculate similarities between frames considered within the local constraints and so at this stage a vast majority of the similarity matrix does not need to be calculated. When the FGP reaches f_W, the window size, the final point $f_W = (m_W, n_W)$ is selected as the end point for the accumulated cost-matrix.

Note that some combinations of constraints that skip points (*i.e.* where i or j are greater than 1) will require that jumps in the FGP are filled in order to compute a complete cost estimate, like in the Type V local constraint, so that the cost estimation of the FGP is complete. A comparison of guidance measures is made in Section 4.

3.5 A-Star Cost Matrix

The windowed area selected is calculated as an accumulated cost matrix between the beginning and end points of the FGP *i.e.* $\mathcal{C}(m, n)$ of $m \in [m_{f_1} : m_{f_L}]$ and $n \in [n_{f_1} : n_{f_L}]$. This accumulated cost matrix can be calculated in either a forward or reverse manner, linking the start to the end point or vice versa. This uses the standard Type I *cost constraint* to determine a frame's accumulated cost as shown by Equation 2:

$$D(m, n) = d(m, n) + \min \begin{Bmatrix} D(m-1, n-1) \\ D(m-1, n) \\ D(m, n-1) \end{Bmatrix} \quad (2)$$

The sub-path $S = (s_1, s_2, ..., s_V)$ is given by the accumulated *cost constraints* by following the cost progression from the beginning point in this window until the hop size is reached. When the sub-path reaches s_V, the final point $f_V = (m_v, n_v)$ is then taken as the starting point for the next window and so on until the end of either sequence is reached. The sub-paths are concatenated to construct the global WTW path. This process can also be seen in Figure 4.

Either of the undirected and directed window end point estimations provide an estimate cost $D(F)$ for each sub-path. This estimate can be used to disregard any points within the accumulated cost matrix that are above this cost

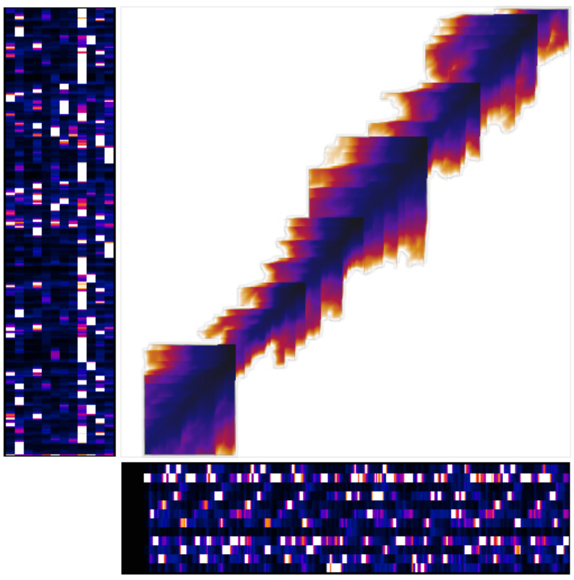

Figure 4. The complete Windowed Time Warping path.

as it is known there is a potential sub-path that is cheaper. The calculation of the similarity for most of these inefficient points can be avoided by calculating the accumulated cost matrix in rows and columns from the end point f_L to the start f_1. When each possible preceding point for the next step of the current row/column has a total cost above the estimated cost *i.e.* $\min(D(m-1,n-1), D(m-1,n), D(m,n-1)) >= D(F)$ the rest of the row/column is then set as more than the cost estimate, thus avoiding calculating the accumulated cost for a portion of the matrix. This procedure can be seen in Figure 5.

4. EXPERIMENTAL EVALUATION

To evaluate WTW we used the score following system with ground truth MIDI, audio and path reference files and compared the accuracy of the found alignments with the known alignments. MATCH, the implementation of On-Line Time Warping [4], was also used to align the test pieces for comparison purposes. In both cases the MIDI was converted to audio using Timidity.

4.1 Mazurka Test Data

The CHARM Mazurka Project by the Centre for the History and Analysis of Recorded Music led by Nick Cook at Royal Holloway, University of London has published a large number of linked metadata files for Mazurka recordings in the form of reverse conducted data, [1] produced by Craig Sapp [12]. We then used template matching to combine this data with MIDI files, establishing links between MIDI notes and reverse conducted notes at the ms level. This provided a set of ground truth files linking the MIDI score to the audio recordings. These ground truths were compared with an off-line DTW alignment and manually supervised to correct any differences found. Overall, 217 sets of audio recordings, MIDI scores and reference files were produced.

[1] http://mazurka.org.uk/info/revcond/

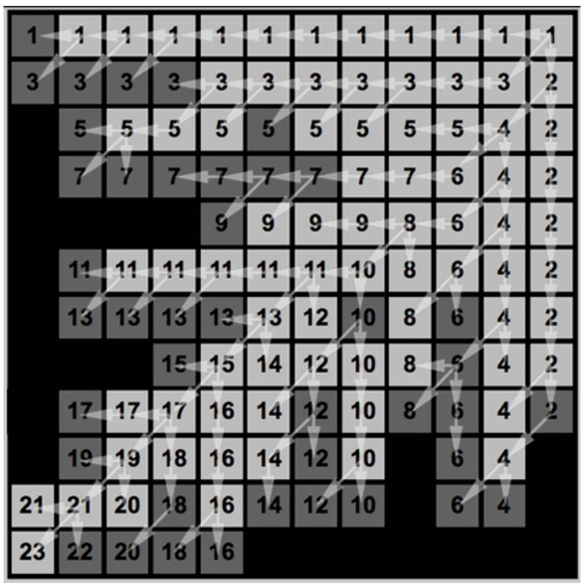

Figure 5. The calculation of the accumulated cost matrix. The numbering shows the order in which rows and columns are calculated and the progression of the path finding algorithm is shown by arrows. Dark squares represent a total cost greater than the estimated path cost whilst black squares indicate points in the accumulated cost matrix that do not need to be calculated.

4.2 Evaluation Metrics

For each path produced by WTW, each estimated audio note time was compared with the reference and the difference was recorded. For differing levels of accuracy requirements (100 ms, 200 ms, 500 ms and 2000 ms), the percentages of notes that were estimated correctly within this requirement for each piece were recorded. These piece-wise accuracies are then averaged for an overall rating. The 2000 ms accuracy requirement is used as the MIREX score following accuracy requirement for notes hit.

4.3 Window Dimensions

The effect of the window size and hop size in WTW is examined in Table 1. The accuracy tests (shown in the top half) show a trend that suggests larger window sizes and smaller hop sizes lead to greater accuracy, as is similar to feature frame dimensions. However, larger window sizes and smaller hop sizes also lead to slower execution times as more points on the similarity matrix were calculated.

4.4 Window Guidance

A comparison of guidance methods for WTW is shown in Table 2. This comparison shows that for the test data used, there was not much difference between directed and undirected WTW and directed only offered an improvement when a large local constraint was used.

Alignment Accuracy at 2000 ms				
	Window Size			
Hop Size	100	200	300	400
100	76.0%	83.1%	81.8%	81.9%
200	63.7%	82.2%	82.0%	82.0%
300	57.7%	77.7%	81.2%	82.5%
400	57.4%	66.1%	81.1%	82.0%

Table 1. Accuracy test results comparing different window and hop sizes for WTW. For this test there was a guidance FGP that used a Type VII +6 *movement constraint* (see Table 2) and the accumulated cost matrix used a Type I *cost constraint* and Type I *movement constraint*.

Alignment Accuracy				
Acc. Req.	100 ms	200 ms	500 ms	2000 ms
None	63.8%	75.9%	82.0%	86.9%
Type I	56.2%	68.7%	74.2%	78.1%
Type IV	63.3%	74.8%	80.7%	86.0%
Type VII	64.0%	76.9%	82.6%	86.6%
Type IV +4	58.0%	70.3%	75.8%	79.5%
Type VII +6	59.4%	72.2%	78.0%	81.2%
Type II	63.9%	75.8%	81.8%	87.3%
Type V	64.9%	78.0%	83.9%	88.1%

Table 2. Accuracy test results comparing different methods for guiding the windows in WTW. The name of the guidance method refers to the *movement constraint* used in the Forward Greedy Path. The 'Type 4 +4' and 'Type 7 +6' constraints include additional horizontal and vertical frames to complete the block. For this test the window and hop size were set at 300ms and the accumulated cost matrix used a Type I *cost constraint* and Type I *movement constraint*.

4.5 Accuracy Results

The results of the accuracy test can be seen in Table 3. From this test we can see WTW produces an accuracy rate comparable with that of OTW. What separates the two methods is that the OTW method took on average 7.38 seconds to align a Mazurka audio and score file where as WTW took 0.09 seconds, (approximately one 80th of the time of OTW). The average length of the Mazurka recordings is 141.3 seconds, therefore, in addition to having the ability to calculate the alignment path sequentially, both methods achieve greater than real-time performance by some margin.

4.6 Efficiency Results

The efficiency tests consisted of aligning sequences of different lengths and recording the execution time. The results of this test can be seen in Table 4. These results show that WTW has linear time costs in relation to the length of the input sequence, unlike standard DTW. The optimisations suggested in this work are shown to decrease the time cost in aligning larger sequences over FastDTW.

Alignment Accuracy				
Acc. Req.	100 ms	200 ms	500 ms	2000 ms
WTW	73.6%	88.8%	94.9%	97.0%
OTW	70.9%	86.7%	94.8%	97.3%

Table 3. Accuracy test results comparing WTW and OTW estimated audio note onset times against references for 217 Mazurka recordings at 4 levels of accuracy requirements. For this test the window and hop size were set at 300ms and the accumulated cost matrix used a Type I *cost constraint* and Type VII *movement constraint*.

Execution time (seconds)				
Sequence length	100	1000	10000	100000
DTW	0.02	0.92	57.45	7969.59
FastDTW (r100)	0.02	0.06	8.42	207.19
WTW	0.002	0.06	0.90	9.52

Table 4. Efficiency test results showing the execution time (in seconds) for 4 different lengths of input sequences (in frames). Results for FastDTW and DTW are from [11]. The r value for FastDTW relates to the radius factor.

5. DISCUSSION AND CONCLUSION

This paper has introduced WTW, a linear cost variation on DTW for real-time synchronisations. WTW breaks down the regular task of creating an accumulated cost matrix between the complete series of input sequence vectors, into small, sequential, cost matrices. Additional optimisations include local constraints in the dynamic programming and cut-off limits for the accumulated cost matrices.

Evaluation of WTW has shown it to be more efficient than state of the art DTW based off-line alignment techniques. WTW has also been shown to match the accuracy of OTW whilst improving on the time taken to process files. Whilst this difference has little effect when synchronising live sequences on standard computers, the greater efficiency of WTW could be useful in running real-time synchronisation methods on less powerful processors, such as those in mobile phones, or when data-mining large data-sets for tasks such as cover song identification.

Future work will involve evaluating WTW on a wider variety of test data-sets, including non-audio related tasks and features. Possible improvements may be found in novel local constraints and/or the dynamic programming used to estimate the start and end points of the accumulated cost matrices. Presently, WTW assumes the alignment is continuous from the start to the end. A more flexible approach will be required to handle alignments made of partial sequence matches. Also, the modifications of WTW could potentially be combined with other modifications of DTW, such as those in FastDTW in order to pool efficiencies. Lastly, WTW, like DTW, is applicable to a number of tasks that involve data-mining, recognition systems or similarity measures. It is hoped WTW makes DTW viable for applications on large data sets in a wide range of fields.

6. ACKNOWLEDGMENTS

This work is supported by the EPSRC project OMRAS2 (EP/ E017614/1). Robert Macrae is supported by an EPSRC DTA Grant.

7. REFERENCES

[1] Bir Bhanu and Xiaoli Zhou. Face recognition from face profile using dynamic time warping. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*, pages 499–502, Washington, DC, USA, 2004. IEEE Computer Society.

[2] David Clifford, Glenn Stone, Ivan Montoliu, Serge Rezzi, François-Pierre Martin, Philippe Guy, Stephen Bruce, and Sunil Kochhar. Alignment using variable penalty dynamic time warping. *Analytical Chemistry*, 81(3):1000–1007, January 2009.

[3] Simon Dixon. Live tracking of musical performances using on-line time warping. In *Proceedings of the 8th International Conference on Digital Audio Effects*, pages 92–97, Madrid, Spain, 2005.

[4] Simon Dixon and Gerhard Widmer. MATCH: A music alignment tool chest. In *Proceedings of the 6th International Conference on Music Information Retrieval*, page 6, 2005.

[5] Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289, New York, NY, USA, 2000. ACM.

[6] Benoít Legrand, C. S. Chang, S. H. Ong, Soek-Ying Neo, and Nallasivam Palanisamy. Chromosome classification using dynamic time warping. *Pattern Recogn. Lett.*, 29(3):215–222, 2008.

[7] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[8] Hiroshi Murase and Rie Sakai. Moving object recognition in eigenspace representation: gait analysis and lip reading. *Pattern Recogn. Lett.*, 17(2):155–162, 1996.

[9] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[10] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1), 1978.

[11] Stan Salvador and Philip Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Workshop on Mining Temporal and Sequential Data*, page 11, 2004.

[12] Craig Sapp. Comparative analysis of multiple musical performances. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR) 2007*, pages 497–500, 2007.

[13] J Serrà, E Gómez, P Herrera, and X Serra. Chroma binary similarity and local alignment applied to cover song identification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(6):1138–1151, 2008.

[14] Robert J. Turetsky and Daniel P.W. Ellis. Ground-truth transcriptions of real music from force-aligned midi syntheses. In *Proceedings of the 4th International Conference on Music Information Retrieval*, 2003.

[15] H. J. L. M. Vullings, M. H. G. Verhaegen, and H. B. Verbruggen. Automated ECG segmentation with dynamic time warping. In *Proceedings of the 20th Annual International Conference of the IEEE In Engineering in Medicine and Biology Society, 1998*, volume 1, pages 163–166, 1998.