GLOBALLY OPTIMAL SHORT-TIME DYNAMIC TIME WARPING APPLICATION TO SCORE TO AUDIO ALIGNMENT

Hagen Kaprykowsky, Xavier Rodet

Ircam – Centre Pompidou – Analyse Synthèse

ABSTRACT

Dynamic Time Warping (DTW) finds the best global match or alignment between two sequences by 'time' warping them optimally. Since the algorithm uses the whole sequences, it can be very demanding in calculation cost and memory. In particular, this limits the size of the sequences which can be aligned. In this paper a novel algorithm is presented, Short-Time Dynamic Time Warping (STDTW), which requires much less memory because optimization is done iteratively on portions of the sequences. The very remarkable property of the algorithm is that under some weak hypothesis, it produces the same globally optimal solution as the classical DTW. As an example, STDTW is applied to Score to Audio Alignment which links events in a musical score and points on the audio performance time axis.

1. INTRODUCTION

Let X and Y be two sequences of elements x_m and y_n and of length M and N respectively:

$$X = x_1, x_2, ...,x_m, ..., x_M = \{x_m, m=1, \ldots, M\} \quad (1)$$

$$Y = y_1, y_2, ...,y_n, ..., y_N = \{y_n, n=1, \ldots, N\} \quad (2)$$

The alignment of an element x_{mk} with an element y_{nk} is defined by a couple $a_k = (m_k, n_k)$, $1 \le m_k \le M$ and $1 \le n_k \le N$.

An alignment of one sequence with the other is defined by a sequence $A = a_1, a_2, ..., a_k, ...,a_K$ (3)

such that the sequences $\{m_k, k=1, \ldots, K\}$ and $\{n_k, k=1, \ldots, K\}$ are non decreasing ($m_{k-1} \le m_k$, $n_{k-1} \le n_k$). If one considers a plane with indices m on the abscissa and n on the ordinate, A also defines a 'path' in this plane.

In Dynamic Time Warping (DTW [1]), indices n and m are called traditionally 'time' because one of the first applications was for speech and the sequences were analysis data at successive time instants. DTW is an algorithm which finds the best alignment (or match), between two (supposedly time) sequences. The alignment is considered as a non-linear 'warping' of one (or the other) sequence along 'time'. To define the best alignment, a (local) distance between an element x_{mk} and an element y_{nk} , $d(x_{mk}, y_{nk})$, is first chosen. To simplify notation, let us denote this distance $d(m_k, n_k)$. Then, the global distance along an alignment or path A is the sum of the weighted local distances along A, i.e. between the sequence elements which are aligned:

$$D(X,Y,A) = \sum_{k=1}^{K} \left(w_k * d\left(m_k, n_k\right)\right), \quad (4)$$

where the w_k are weights to be defined in the following.

The optimal alignment A_O is the one which minimizes the global distance $D(X, Y, A)$.

Score to Audio Alignment links events in a score and points on the audio performance time axis. The audio performance is a digital recording (a signal) of the score played by musicians, which is referred to as the performance. Here the score is represented by a Standard MIDI File (SMF). An alignment links notes of the score with their position in the performance, typically the beginning and end of the notes. This is not a trivial task since the exact tempo is not known and not constant, the notes are never played exactly as written in the score, and finding notes in a complex polyphonic performance is extremely difficult. The performance audio signal can be coded as a sequence of frequency analysis data along time, and the score is a sequence of notes and chords. Therefore, DTW provides a globally optimal alignment of the score to the recording of a performance. But DTW computes the optimal path on the entire sequences and thus necessitates a huge amount of memory. As an example, suppose the duration of the performance is 15'. For a very precise alignment, the frequency analysis should be done approximately every 2.5 ms. This amounts to a sequence of 360000 elements. If there is a mean of 8 notes or chords per second, this leads to a sequence of 7200 elements. Then the usual DTW algorithm requires arrays of size 180000*7200=2.592 G cells.

However, it is clear that the entire score and the entire performance are not necessary to perform a correct alignment. Most of the time, a shorter portion is enough, provided that it contains a sufficient number of notes and that it cannot be mistaken for another (unambiguity). There are short-time versions of DTW that can be applied iteratively on shorter portions [2]. But the limits of these portions in the score and in the performance are obviously not known since the alignment is not known. Also, while DTW finds the globally optimal alignment, these short-time versions do not guarantee that the short-time solution is the globally optimal one.

To fulfill these requirements, we have designed a new algorithm, called STDTW (Short-Time Dynamic Time Warping), which relies on a short-time alignment and, under some weak hypothesis, provides the same globally optimal solution as the classical global DTW algorithm.

STDTW iteratively works on a short-time portion of the sequences and finds, however, at each iteration a portion of the optimal path. This means that, after each iteration, the computation can be stopped, the path stored and the algorithm re-started from the last point of the path, i.e. on a portion of the two sequences as if they where new sequences.

In other words, due to this nice property, the algorithm can be re-initialized at a point of the optimal alignment path which has been determined before. This then allows the alignment of arbitrarily long sequences. In case of an interruption in the computation (e.g. system crash), the algorithm can be continued from the last point stored without having to start from the very beginning.

As mentioned above, the portions that can be aligned should be unambiguous. More precisely, theses portions are found automatically by the algorithm. This means that the STDTW automatically finds the shortest portions which are unambiguous, hereby providing some valuable information about the structure of the sequences.

Finally, within a 'real time' context, STDTW provides the alignment result with the shortest possible delay according to unambiguity; this is as if one had 'heard' enough of the audio file (or whatever sequence) to be certain of the corresponding portion

of score (or sequence). A well-known real time alignment approach, more commonly known as Score-Following which uses Dynamic Programming, is presented in [3].

2. DYNAMIC TIME WARPING

Let us first briefly review the classical DTW algorithm. Let X and Y be two sequences of elements x_m and y_n and of length M and N respectively (1), (2). Let $d(x_{mk}, y_{nk})$ be the *local* distance between an element x_{mk} and an element y_{nk}. To simplify the notation, let us write this distance $d(m_k, n_k)$. An alignment of one sequence with the other is defined by a sequence, or path, A (3). The global distance along the path A is the sum of the weighted local distances along A, i.e. the distances between the sequence elements which are aligned (4).

2.1. Path Constraints

Possible paths A are limited by several constraints.

Endpoint Constraints force the warping path to start and finish in the opposite diagonal corner of the rectangle *(m,n)*.

$a_1 = (1,1)$ (5)

$a_K = (M,N)$ (6)

Monotonicity Constraints force the points a_k to be monotonically placed in time.

If $a_k = (m_k, n_k)$ and $a_{k-1} = (m_{k-1}, n_{k-1})$ (7)

then $m_k - m_{k-1} \geq 0$ and $n_k - n_{k-1} \geq 0$ (8)

Local Continuity Constraints force the possible steps, in the warping path, to adjacent cells (including diagonally adjacent cells).

If $a_k = (m_k, n_k)$ and $a_{k-1} = (m_{k-1}, n_{k-1})$ (9)

then $m_k - m_{k-1} \leq 1$ and $n_k - n_{k-1} \leq 1$ (10)

The DTW algorithm first calculates the augmented distance array *adm* of size MxN where *adm(m,n)* is the cost of the best path from (1,1) up to *(m,n)*. The value *adm(m,n)* is computed recursively by use of the local distance *d(m,n)*, of the weights *w* and of the values *adm(i,j)* in a neighborhood (*i,j*) of *(m,n)* with i<m and j<n. The points in this neighborhood are called the *predecessors* of *(m,n)* (see figure 1).

There exist different types of local constraints and the weights along the local path branches can be tuned in one direction or another if needed. These weights [w_v w_h w_d] are explained in figure 1. The different type names Type I, III and V follow the notation in [1].

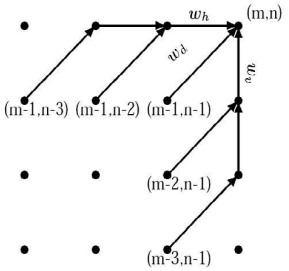

Fig. 1 Local path constraint - Type V

In case of Type V, the accumulated distance to any point *(m,n)* is calculated recursively as in equation (11), with *d(m,n)* abbreviated to λ. For Score to Audio Alignment, Type V was found to be the best [4]. Type V constraints the mean slope of path A to be between 3 and 1/3. Indeed, a performance very rarely has tempo variations greater than 3 or less than 1/3 with respect to the mean tempo. Therefore, Type V provides good alignment while avoiding vertical or horizontal paths. The standard values for the local path weights [w_v w_h w_d] are [1 1 2] for Type V. Our experiments showed that lowering w_d is better since it favors the diagonal and prevents extreme slopes [4].

$$adm(m,n) = \min \left\{ \begin{array}{l} adm(m-1,n-1) + w_d\,\lambda \\ adm(m-2,n-1) + w_v\,\lambda \\ \quad +w_d\,d(m-1,n) \\ adm(m-1,n-2) + w_h\,\lambda \\ \quad +w_d\,d(m,n-1) \\ adm(m-3,n-1) + w_v\,\lambda \\ \quad +w_d\,d(m-2,n) \\ \quad +w_v\,d(m-1,n) \\ adm(m-1,n-3) + w_h\,\lambda \\ \quad +w_h\,d(m,n-1) \\ \quad +w_d\,d(m,n-2) \end{array} \right\} \qquad (11)$$

2.2. Dynamic programming

There are exponentially many warping paths that satisfy the above conditions. Dynamic programming is a fast way to find the optimal path, which minimizes the global distance, i.e. the total warping cost *adm(M,N)*. It relies on the Viterbi algorithm with complexity $O(n^2)$ applied on the *adm* array. This is done in a backtrack final step that very simply computes optimal predecessors from *(M,N)* to (1,1) (see [1] for details).

From now on we will only consider paths which are optimal from (1,1) to a certain *(m,n)*, and we call them paths without mentioning this restriction.

2.3. Range and Memory Reductions

2.3.1. Global path constraints

Type V constrains the slope to be between 3 and 1/3. Using this slope constraint leads to a reduced warping window which is sometimes referred to the Itakura Parallelogram.

2.3.2. Path Pruning

In our experiments, to reduce the computation time and the resources needed, at every iteration *m* we keep only the best paths by pruning the paths which have an augmented distance *adm(m,n)* greater than a given threshold θ_p. This threshold is dynamically set using the minimum of the previous *adm* row. Global path constraints and Path Pruning finally define a *corridor* (see figure 3) which limits the possible paths.

2.3.3. Shortcut Path

In some cases, one is interested only in the alignment of note-onset times and not by the evolution within a note. In such cases, one only needs to keep in memory all the corresponding shortcut paths, as presented in [4].

3. LIMITS

Even if calculation and memory costs have been reduced by using the Range and Memory Reductions presented in section 2.3, we found that the alignment was only feasible for polyphonic pieces of approximately 5 minutes and 5000 notes maximum. The main problem is the memory cost, since all predecessors have to be stored for calculating the final backtrack from *(M,N)* to (1,1) to obtain the globally optimal warping path.

4. SHORT-TIME DYNAMIC TIME WARPING

In general it is not possible to find the globally optimal alignment without calculating all accumulated distances and saving all corresponding predecessors. The problem is now to replace the global optimization for the entire sequence by a short-time, i.e. local, optimization, and obtain the same solution.

4.1. Suboptimal Approaches

There have been many approaches for a short-time alignment in the past, particularly in speech recognition [2]. To determine the short-time alignment, a smaller window is slid from the beginning to the end, obtaining only approximately the same solution as the global one.
By using a smaller window, the problem of relaxation of the endpoint constraints has to be solved. The main problem is, that the short-time alignment will be in general suboptimal if the path is not near the diagonal [2].

4.2. Optimal Approach

With respect to the initial constraints, it is obvious that all paths have to have a common point at least at (1,1). In figure 3 one can see all backtracks in the corridor from a given m (audio frame index). There appear to be a portion common to all paths from point (1,1) up to the so called *fusion point* where all paths fuse. There is only one path which goes from (1,1) to the fusion point. This path is therefore a part of the optimal path from *(M,N)* to (1,1), so it is globally optimal.
Now suppose one path can not cross an other one. Then it is sufficient to only calculate the two backtracks from r_{min} and r_{max} (left and right corner of the corridor, figure 3) to determine the fusion point and obtain a part of the global optimal path from (1,1) to the fusion point. For Type I it is clear that paths can not cross each other (principle of optimality [5]). For Type V this does not seem obvious. From now on, let us now suppose that
$[w_v\ w_h\ w_d] = [1\ 1\ 1]$.

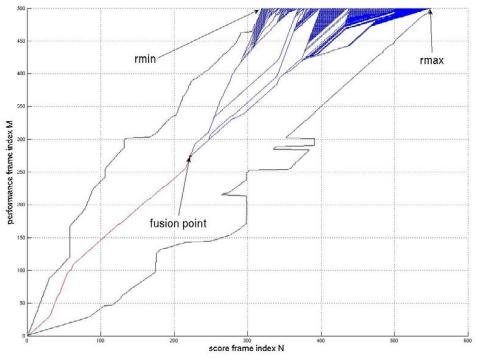

Fig. 3 All backtracks inside the corridor

4.2.1. No crossing paths with Type V
Suppose one knows an optimal backtrack path from the point *(M,N)* to the point *(m,n)*. We then want to prove that this path can not cross another optimal path at any of the direct predecessors of *(m,n)*, i.e. those below and left of *(m, n)* (figure 1). *adm(m,n)* is defined recursively by equation (11). It is obvious that the paths won't cross, neither on *(m,n),* nor on any of the predecessors of *(m,n)* (principle of optimality [5]). But this does not prove that a crossing cannot occur at other points on horizontal line m or vertical line n. To prove this, let us separate the paths, between *(m,n)* and the direct predecessors, into two classes, class 1 and class 2 (figure 5).

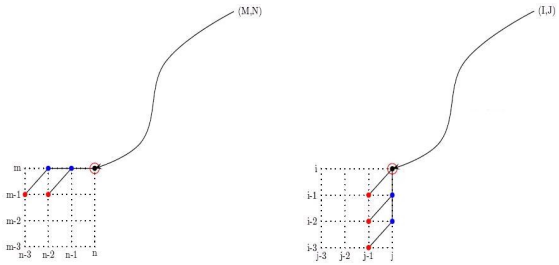

Fig. 5 Class 1 and Class 2

The paths of class 1 can only be crossed at the points *(m,n-1)* or *(m,m-2)*. Let us show that they cannot be crossed by a path of class 2. If this would happen, then the path from *(i,j)* to one of the predecessors in class 2 *(j-1,i-1)*, *(j-1,i-2)*, *(j-1,i-3)* would cross the path issued from *(m,n)* in one of the predecessors of class 1. It is sufficient to examine all the above mentioned possible crossings, one for one. To simplify notation, we write $p_{k,l}$ for local distance *d(m-k,n-l)* and $\bar{p}_{k,l}$ for accumulated *adm(m-k,n-l)*. The local distance of the crossing point is noted *C*. We only show the two pairs of inequalities (12), (13) and (15), (16) which result immediately from figures 6 and 7. The other cases lead to the same result as the inequalities (12), (13) of figure 6.
Each pair of inequalities (with the exception of the crossing of figure 7) simplifies to $C = 0$ (14). This means that a crossing is only possible if the local distance at the crossing point is zero. This is excluded in our case because it is (in a probabilistic sense) "nearly impossible" that this distance is zero. The pair of inequalities (15), (16) of figure 7 can be simplified to an inequality $\bar{p}_{2,3} = p_{2,4} + \bar{p}_{1,3}$ (17) which is also "nearly impossible". The complete proof is illustrated in detail in [6].

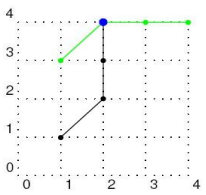

Fig. 6

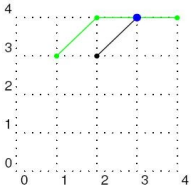

Fig. 7

$$p_{4,4} + p_{3,4} + C + \bar{p}_{1,3} \leq p_{4,4} + p_{3,4} + \underbrace{p_{2,3} + p_{2,2} + \bar{p}_{1,1}}_{S} \quad (12)$$

$$C + \underbrace{p_{2,3} + p_{2,2} + \bar{p}_{1,1}}_{S} \leq C + \bar{p}_{1,3} \quad (13)$$

$$\begin{aligned} C + \bar{p}_{1,3} &\leq S \leq \bar{p}_{1,3} \\ \Rightarrow C &= 0 \end{aligned} \quad (14)$$

$$p_{4,4} + C + p_{2,4} + \bar{p}_{1,3} \leq p_{4,4} + C + \bar{p}_{2,3} \quad (15)$$

$$C + \bar{p}_{2,3} \leq C + p_{2,4} + \bar{p}_{1,3} \quad (16)$$

$$\begin{aligned} p_{2,4} + \bar{p}_{1,3} &\leq \bar{p}_{2,3} \\ \bar{p}_{2,3} &\leq p_{2,4} + \bar{p}_{1,3} \\ \Rightarrow \bar{p}_{2,3} &= p_{2,4} + \bar{p}_{1,3} \end{aligned} \quad (17)$$

5. METHOD

In order to determine a part of the global optimal alignment it is sufficient to determine the fusion point of the two backtracks

(figure 8: backtrack 1 and backtrack 2). The path from (1,1) to the fusion point is a part of the optimal path, because the backtracks can not cross each other as proved here above.
The algorithm proceeds with increasing values of one of the indices, say m=1, 2, 3, …. . In order to diminish the cost of two backtracks and of the fusion point determination, these are done only every mStep, e.g. mStep = 100, by the modulo instruction here below.

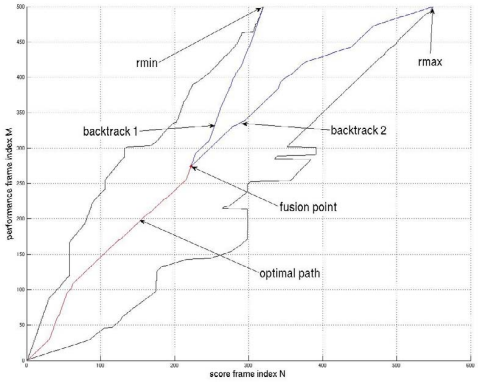

Fig. 8 The two backtracks from r_{min} and r_{max}

Algorithm

For m=1 to M

- Calculation of *adm(m,n)* and storage of the optimal predecessor of *(m,n)*.

If (modulo(m, mStep)) = 0

- Calculation of the backtracks from r_{min} and r_{max} and determination of the fusion point.
- At this point it is only necessary to keep the predecessors inside the last backtracks and the part of the optimal path which has been determined until the fusion point. All the previous data can be cleared.

If (memory cost to high)

- reinitialisation
- calculation of the short cut path
- reduction of the threshold of the path pruning

endFor

6. PERFORMANCE

The new Short-Time Dynamic Time Warping algorithm theoretically permits the alignment of a performance of arbitrary length, and to obtain the same globally optimal result as the global DTW algorithm.
It is possible to reinitialise the algorithm at the last fusion point which has already been determined. This gives the possibility to stop the algorithm, as well as to restart the calculation (if some system problem occurred for instance) without having to recalculate the whole alignment. Note that restarting the algorithm at a fusion point is the same procedure as starting from the beginning of a new sequence (the fusion point becomes the beginning point (1,1) of the new sequences to be aligned).
Furthermore, fusion points f_p have other very interesting properties. First, the common path portion from (1,1) to f_p is a part of the globally optimal path from (1,1) to *(M,N)*. Suppose one starts from f_p. Then f_{p+1} is obviously the first next point such that the optimal alignment can be insured locally between f_p and f_{p+1}, i.e. when looking only at the data between f_p and f_{p+1}. That is, f_{p+1} is the first next point such that the two sequences contain enough information for the alignment to be unambiguous between f_p and f_{p+1}. Therefore, fusion points f_p provide some interesting information about the structure of the sequences: they mark the segments which are unambiguous enough to be certainly aligned. For instance, this is easily observed in Score to Audio Alignment: new fusion points occur when there have been enough notes in the score since the previous fusion point for this part of the performance to be certainly attributed to the corresponding part of the score. By using the classical DTW algorithm the alignment is only possible for (approximately) polyphonic pieces of maximum 5 minutes or 5000 notes respectively, because of the high memory costs. We implemented the new STDTW algorithm in MATLAB with a MEX library which has been developed at Ircam. On a computer with 2 gigabyte of Ram, Jazz and Classic pieces up to 10 minutes have been aligned. Since the Short Cut path has not yet been implemented, all predecessors inside the backtracks need to be stored. Better performance will be obtained when the Short Cut path will be implemented.

7. CONCLUSION

A new Short-Time DTW algorithm has been designed and implemented. It theoretically permits to align pieces of arbitrary length while providing the same results as the global DTW algorithm. It can easily be applied on portions only of long pieces. It can also easily be restarted from an intermediate point without having to recompute the whole sequence alignment. Also, it provides an interesting insight into the structure of the sequences to be aligned.

8. FUTURE WORKS

The Short Cut Path will be implemented in the program, thus allowing an even better performance, in particular aligning longer pieces and faster. The new algorithm could be envisioned for Score-Following.

Acknowledgments

D.Schwarz, F. Soulez, J. Escribe, G. Peeters, P. Bernat, N. Orio.

10. REFERENCES

[1] L. R. Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, 1993.

[2] J. Di Martino, “Contribution à la reconnaissance globale de la parole: mots isolés et mots enchaînés,” PhD Thesis, Nancy, France, 1984.

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

[4] F. Soulez, X. Rodet and D. Schwarz, “Improving Polyphonic and Poly-Instrumental Music to Score Alignment,” International Symposium on Music Information Retrieval (ISMIR), Baltimore, USA, 2003.

[5] R. Bellman, *Dynamic Programing*, Princeton University Press, 1957.

[6] H. Kaprykowsky, “Alignement d’un enregistrement Audio avec sa Partition : passage de l’algorithme DTW global à un DTW à court terme,” Master Thesis, Ircam Centre Pompidou, Paris, France, 2005.