Coarse-DTW: Exploiting Sparsity in Gesture Time Series

Marc Dupont[1,2] and Pierre-François Marteau[1]

[1] IRISA, Université de Bretagne Sud, Campus de Tohannic, Vannes, France
[2] Thales Optronique, 2 Avenue Gay Lussac, Elancourt, France

Abstract. Dynamic Time Warping (DTW) is considered as a robust measure to compare numerical time series when some *time elasticity* is required. Even though its initial formulation can be slow, extensive research has been conducted to speed up the calculations. However, those optimizations are not always available for multidimensional time series. In this paper, we focus on time series describing gesture movement, all of which are multidimensional. Our approch propose to speed up the processing by 1. adaptively downsampling the time series into *sparse* time series and 2. generalizing DTW into a version exploiting sparsity. Furthermore, the downsampling algorithm doesn't need to know the whole timeseries to function, making it a good candidate for streaming applications such as real-time gesture recognition.

1 Introduction

Among other measures, Dynamic Time Warping (DTW) has been widely popularized during the seventies with the advent of speech recognition systems [18], [14]. However, one of the main drawbacks of such a *time-elastic* measure is its quadratic computational complexity which, as is, prevents processing a very large amount of lengthy temporal data. Recent research has thus mainly focused on circumventing this complexity barrier. The original approach proposed in this paper is to cope directly and explicitly with the potential sparsity of the time series during their *time-elastic* alignment.

2 Previous work

DTW has seen speed enhancements in several forms; [14] and [5] reduce the search space by using a band or parallelogram; [1] introduced the concept of a sparse alignment matrix to dynamically reduce the search space without optimality loss. The dimensionality of the data can be reduced, such as in [21] and [7] who propose Piecewise Aggregate Approximation (PAA) to downsample the time series into segments of constant size, then handled by a DTW modification, PDTW [9]; further compressing can be obtained with Adaptive Piecewise Constant Approximation (APCA) [2]; or compression via symbolic representation of scalar points can be obtained with SAX [13]. Early abandoning strategies

avoid useless calculation by computing cheap lower bounds: such as [20], [8] and [10], but the most powerful [8] is not readily available in a form available for multidimensional time series. ID-DTW (Iterative Deepening DTW) [4] and FastDTW [16] use multi-resolution approximations, possibly with an early abandoning strategy; [15] and [17] have also proposed approaches mixing APCA with a lower bounding strategy. Additionally, some alternative elastic distance variants have been proposed, such as ERP [3] or TWED [12] with some gain in classification accuracy, but with no speed-up strategy designed so far.

Our method differs from the previous work as follows: first, it gives a novel way of producing a piecewise constant time series, especially interesting because of its simplicity and its potential use in streaming scenarios (the downsampled time series is produced as fast as the original one arrives); second, DTW is enhanced with a new weighting strategy to accept such downsampled time series and achieve the desired speed enhancement.

3 Presentation of Coarse-DTW

3.1 Sparse time series

The usual notion of a time series will be called here a *dense time series*. It represents a sequence (v_i) of points in $\mathbb{R}^d$, where d is the dimension. Such a time series is usually sampled at a regular interval.

By contrast, let a *sparse time series* be a pair of sequences (s_i) and (v_i) with the same length n:

$$\begin{aligned} s &: \{1, \ldots, n\} \to \mathbb{R}_+ \\ v &: \{1, \ldots, n\} \to \mathbb{R}^d \end{aligned} \tag{1}$$

Each v_i represents a multidimensional point (of dimension d) and each s_i is a number describing *how long the value v_i lasts*. We call this number s_i the *stay* of v_i. In the following, we will also denote a sparse time series as $\{(s_1, v_1), \ldots, (s_n, v_n)\}$.

For example, the 2D dense time series $\{(0.5, 1.2), (0.5, 1.2), (0.3, 1.5)\}$ is equivalent to the 2D sparse time series $\{(2, (0.5, 1.2)), (1, (0.3, 1.5))\}$. As another example, a dense time series (v_i), is exactly represented by the sparse time series with the same values v_i and all stays $s_i = 1$.

3.2 Coarse-DTW

The Coarse-DTW algorithm accepts two sparse time series: (s_i, v_i) of length n, and (t_j, w_j) of length m.

Algorithm 1 Coarse-DTW

```
1: procedure COARSE-DTW((s, v), (t, w))
2:     A = new matrix [0..n, 0..m]
3:     A[0, .] = A[., 0] = ∞ and A[0, 0] = 0
4:     for i = 1 to n do
5:         for j = 1 to m do
6:             A[i, j] = min( s_i.δ(v_i, w_j) + A[i−1, j],
7:                            t_j.δ(v_i, w_j) + A[i, j−1],
8:                            max(s_i, t_j).δ(v_i, w_j) + A[i−1, j−1] )
9:     return A[n, m]
```

The symbol δ represents any distance on $\mathbb{R}^d$. A common choice is $\delta(x, y) = \|x - y\|_2^2 = \sum_{k=1}^{d}(x_k - y_k)^2$.

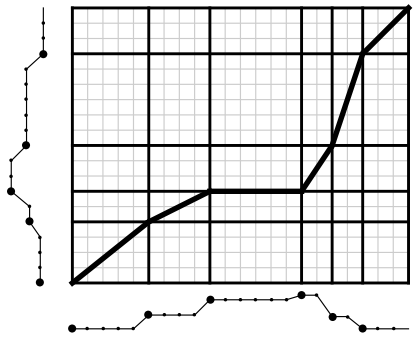

Fig. 1. A warping path in Coarse-DTW. We superimposed the sparse timeseries (bigger points) on top of their equivalent dense timeseries (smaller points). The coarse, thick grid is the Coarse-DTW matrix, whereas the underlying thin grid is the classical DTW cost matrix.

Coarse-DTW takes advantage of the sparsity in the time series to calculate costs efficiently. However, because the points last for different amount of time, we must adapt the classical DTW formulation in order to account for the stays s_i and t_j of each point into the aggregate cost calculation.

Obviously, when a point lasts for a long time, it should cost more than a point which lasts for a brief amount of time. For this reason, the pure cost $\delta(v_i, w_j)$ is multiplied by some quantity, called *weight*, linked to how long the points last, as in lines 6–8 of the algorithm. The goal of this subsection is to explain why we set those weights to s_i, t_j, and $\max(s_i, t_j)$ respectively.

The choice of weights s_i and t_j in lines 6 and 7 is motivated as follows: when we advance one time series without advancing the other, we want a lengthy point to cost more than a brief point. In the DTW constant-cost sub-rectangle, advancing the first time series is like following an horizontal subpath, whose aggregated cost would be $\delta(v_i, w_j)$ on each of its s_i cells. This sums up to $s_i.\delta(v_i, w_j)$, which

is why the weight is chosen to be s_i in line 6. An analog interpretation holds for a vertical subpath of t_j cells.

In a constant-cost sub-rectangle (of size $s_i \times t_j$), minimizing the aggregated cost of a path is equivalent with minimizing its number of cells, because all cells have the same cost. Furthermore, the minimal number of cells is exactly $\max(s_i, t_j)$. This would be the path followed by classical DTW. Hence, the weight is set to $\max(s_i, t_j)$ in line 8.

4 Downsampling

In this section, we seek to transform a dense time series (u_i) into a sparse time series (s_i, v_i); the goal is to detect when series "move a lot" and "are rather static", adjusting the number of emitted points accordingly.

Bubble downsampling can be described in a simple form as follows:

Algorithm 2 Bubble Downsampling

```
1:  procedure BUBBLE(v, ρ)                                  ▷ ρ ≥ 0
2:      i_center = 1                                        ▷ initialize bubble center
3:      v_center = v_1
4:      v_mean = v_1
5:      for i = 2 to n do
6:          Δv = δ(v_i, v_center)                           ▷ distance to center
7:          Δi = i − i_center                               ▷ find the stay
8:          if Δv ≥ ρ then                                  ▷ does the bubble "burst"?
9:              yield (Δi, v_mean)                          ▷ emit stay + point
10:             i_center = i                                ▷ update bubble center
11:             v_center = v_i
12:             v_mean = v_i
13:         else
14:             v_mean = (Δi × v_mean + v_i)/(Δi + 1)       ▷ update mean
15:     Δi = n − i_center + 1                               ▷ force bursting last bubble
16:     yield (Δi, v_mean)
```

The idea behind Bubble downsampling is based on the following approximation: consecutive values can be considered equal as long as they stay within a given radius ρ for the distance δ. We can picture a curve which makes bubbles along its path (see Fig. 3), hence the name. Concretely, the algorithm emits a sparse time series, where each stay is the number of consecutive points contained in a given bubble, and each value is the mean of the points in this bubble.

The parameter ρ represents the tradeoff between information loss and density. A large ρ emits few points, thus yielding a very sparse time series, but less accurate; a smaller ρ preserves more information at the expense of a lower downsampling ratio. The degenerate case $\rho = 0$ will output a clone of the original time series with no downsampling (all stays equal to 1). Because speed is a direct consequence of sparsity in Coarse-DTW, a good middle value for ρ must

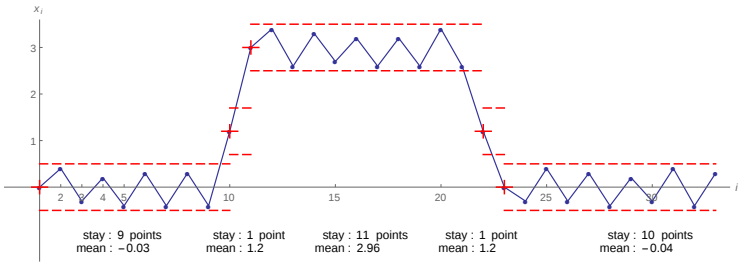

Fig. 2. Bubble downsampling applied on a 1D time series (blue, solid) with $\rho = 0.5$. The 1-bubbles are represented by their 1-centers (red crosses) and their 1-boundaries (red, dashed lines). The sparse time series emitted is $\{(9, -0.03), (1, 1.2), (11, 2.96), (1, 1.2), (10, -0.04)\}$.

be found, so that time series are as sparse as possible while retaining just the right amount of information.

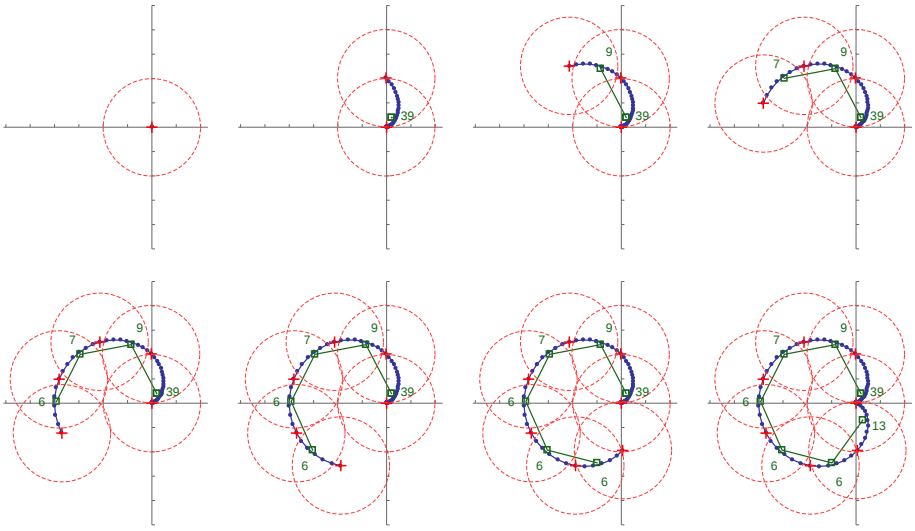

Fig. 3. Bubble downsampling progressively applied on a 2D time series (outer blue line with dots) with $\rho = 2.0$, along with the sparse time series emitted (inner green line with squares). Again, the 2-bubbles are represented by their 2-centers and their 2-boundaries (red crosses and dashed circles). Numbers indicate the stays. Notice how stays take into account the slowness at the beginning of the signal.

5 Optimizations on Coarse-DTW

DTW suffers from a slow computation time if not implemented wisely. For this reason, several optimizations have been designed [8]. The next optimizations we

considered are called *lower bounds*, designed to early-abandon computations in a k-Nearest Neighbor scenario.

The first lower bound $\mathrm{LB}_{\mathrm{Kim}}$ [20] is transposable to Coarse-DTW: as with 1D time series, the first and last pairs of points will always be matched together as long as the timeseries have each at least two points. First, the cost of matching the first points is: $\max(s_1, t_1).\delta(v_1, w_1)$ because the first matching is done diagonally ($A[0,1] = A[1,0] = \infty$). Then, the cost of matching the last points is $\min(s_n, t_m, \max(s_n, t_m)) \,.\, \delta(v_n, w_m)$. Hence, the lower bound is written:

$$\text{Coarse-DTW}(v, w) \geq \max(s_1, t_1).\delta(v_1, w_1) + \min(s_n, t_m, \max(s_n, t_m)).\delta(v_n, w_m) \tag{2}$$

The second lower bound can be evaluated several times as DTW progresses: for any row i, the minimum of all cells $A[i,.]$ is a lower bound to the DTW result. Indeed, this result is the last cell of the last row, and the sequence mapping a row i to $\min_j A[i,j]$ is increasing, because the costs are positive. Hence, during each outer loop iteration (i.e., on index i), we can store the minimum of the current row and compare it to the best-so-far for possibly early abandoning. This can be transposed directly to Coarse-DTW without additional modifications.

Finally, probably the most powerful lower bound for unidimensional time-series, known as $\mathrm{LB}_{\mathrm{Keogh}}$ [8], is based upon the calculation of an envelope; however this calculation is not trivially transferable to the case of multidimensional time series simply by generalizing the uni-dimensional equations. Thus, we will unfortunately not consider it in our study.

6 Results

6.1 DTW vs. Coarse-DTW in 1-NN classification

We considered the classification accuracy and speed of three multidimensional labeled time series datasets describing gesture movement. The classifier is 1-NN and we enabled all optimizations described earlier that apply to multidimensional time series, namely: $\mathrm{LB}_{\mathrm{Kim}}$ and early abandoning on the minima of rows. We report only the classification time; learning time is zero because no processing is required. Each dataset is run once with DTW and several times with Coarse-DTW, each time with a different value for the downsampling radius ρ.

MSRAction3D [11] time series have 60 dimensions (twenty 3D joints) which we classified by cross-validating all 252 combinations of 5 actors in training and 5 in test. uWaveGestureLibrary_[XYZ] comes from the UCR time series database [6]; it can be considered as three independent uni-dimensional datasets, but we rather used it here as a single set of 3-dimensional time series, which makes 1-NN DTW classification fall from 1D errors of respectively 27.3 %, 36.6 % and 34.2 % down to only 2.8 % as a 3D time series. Character Trajectories [19] comes from the UCI database and describes trajectories of character handwriting; they were first resampled to have all the same size (204 data points, size of the longest sequence in the dataset).

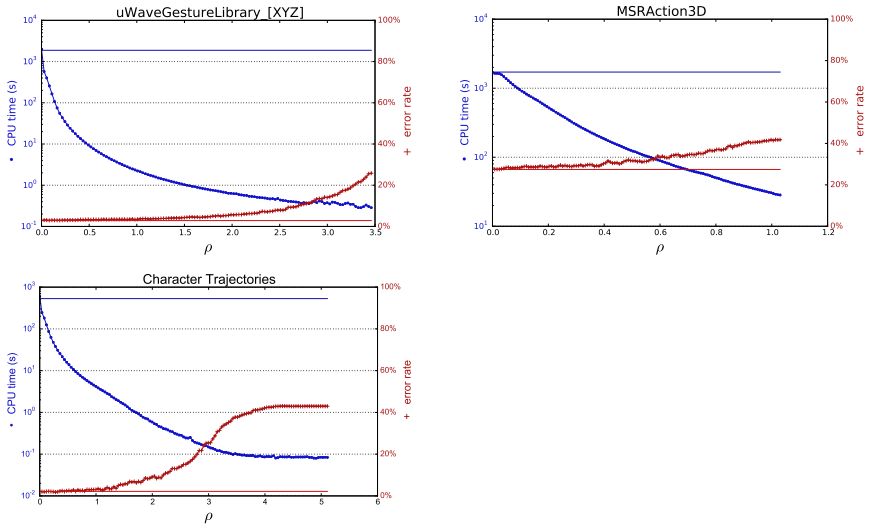

Fig. 4. 1-NN classification time and error rate of Coarse-DTW as ρ increases. For comparison, DTW results are shown as horizontal bars (independent of ρ). Speedups from 10x to 1000x are obtained without sensible accuracy degradation.

7 Conclusions

Not only have we transposed DTW into Coarse-DTW, a version accepting sparse time series, but we have also developed Bubble, an extremely efficient algorithm to generate such sparse time series from regular ones. By coupling those two mechanisms, we found out that time series can be classified much faster in nearest-neighbor classification; the user can reach the desired tradeoff between speed and accuracy, by tuning the parameter ρ in the downsampling algorithm. Gesture timeseries produce smooth time series which present a considerable ability to be downsampled, producing good results in classification speedup.

In order to learn ρ from the data, experiments above suggest a simple method: first, set an acceptable threshold on the error (e.g. +2% w.r.t. regular DTW error) ; then, select the ρ whose error is under this threshold and classification speed is fastest.

Although we didn't cover it in our test scenarios, it is worth highlighting that Coarse-DTW and Bubble are directly applicable to a streaming scenario: indeed, Bubble doesn't need to know the whole timeseries before emitting sparse points. As a consequence, it could be a great way to save CPU time and battery life in an embedded gesture recognition setup.

8 Acknowledgements

This study was co-funded by the ANRT agency and Thales Optronique SAS, under the CIFRE convention 2013/0932.

References

1. Ghazi Al-Naymat, Sanjay Chawla, and Javid Taheri. Sparsedtw: A novel approach to speed up dynamic time warping. In *Proceedings of the Eighth Australasian Data Mining Conference - Volume 101*, AusDM '09, pages 117–127, Darlinghurst, Australia, Australia, 2009. Australian Computer Society, Inc.
2. Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, June 2002.
3. L. Chen and R. Ng. On the marriage of lp-norm and edit distance. In *Proceedings of the 30th International Conference on Very Large Data Bases*, pages 792–801, 2004.
4. Selina Chu, Eamonn J. Keogh, David M. Hart, and Michael J. Pazzani. Iterative deepening dynamic time warping for time series. In Robert L. Grossman, Jiawei Han, Vipin Kumar, Heikki Mannila, and Rajeev Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, USA, April 11-13, 2002*, pages 195–212. SIAM, 2002.
5. F. Itakura. Minimum prediction residual principle applied to speech recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 23(1):67–72, Feb 1975.
6. E. J. Keogh, X. Xi, L. Wei, and C.A. Ratanamahatana. The UCR time series classification-clustering datasets, 2006. http://wwwcs.ucr.edu/ eamonn/time_series_data/.
7. Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *JOURNAL OF KNOWLEDGE AND INFORMATION SYSTEMS*, 3:263–286, 2000.
8. Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3):358–386, March 2005.
9. 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*, KDD '00, pages 285–289, New York, NY, USA, 2000. ACM.
10. Daniel Lemire. Faster retrieval with a two-pass dynamic-time-warping lower bound. *Pattern Recognition*, 42(9):2169 – 2180, 2009.
11. W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In IEEE CS Press, editor, *Proc. IEEE Int'l Workshop on CVPR for Hum. Comm. Behav. Analysis*, pages 9–14, 2010.
12. P. F. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):306–318, 2008.
13. Pranav Patel, Eamonn Keogh, Jessica Lin, and Stefano Lonardi. Mining motifs in massive time series databases. In *In Proceedings of IEEE International Conference on Data Mining (ICDM'02*, pages 370–377, 2002.
14. H. Sakoe and S. Chiba. A dynamic programming approach to continuous speech recognition. In *Proceedings of the 7th International Congress of Acoustic*, pages 65–68, 1971.
15. Yasushi Sakurai, Masatoshi Yoshikawa, and Christos Faloutsos. Ftw: Fast similarity search under the time warping distance. In *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, pages 326–337, New York, NY, USA, 2005. ACM.

16. Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, October 2007.
17. Yutao Shou, Nikos Mamoulis, and David W. Cheung. Fast and exact warping of time series using adaptive segmental approximations. *Mach. Learn.*, 58(2-3):231–267, February 2005.
18. V. M. Velichko and N. G. Zagoruyko. Automatic recognition of 200 words. *International Journal of Man-Machine Studies*, 2:223–234, 1970.
19. Ben H Williams, Marc Toussaint, and Amos J Storkey. *Extracting motion primitives from natural handwriting data*. Springer, 2006.
20. Sang wook Kim, Sanghyun Park, and Wesley W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *In ICDE*, pages 607–614, 2001.
21. Byoung-Kee Yi and Christos Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 385–394, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.