Warping the Time on Data Streams

Paolo Capitani[1] **, Paolo Ciaccia**[1]

[1]DEIS - IEIIT-BO/CNR, University of Bologna, Italy

{pcapitani,pciaccia}@deis.unibo.it

Abstract. *Continuously monitoring through time the correlation/distance of multiple data streams is of interest in a variety of applications, including financial analysis, video surveillance, and mining of biological data. However, distance measures commonly adopted for comparing time series, such as Euclidean and Dynamic Time Warping (DTW), either are known to be inaccurate or are too time-consuming to be applied in a streaming environment. In this paper we propose a novel DTW-like distance measure, called $SDTW$, which, unlike DTW, can be efficiently updated at each time step and experimentally show that it improves over DTW by orders of magnitude without sacrificing accuracy. For instance, with a sliding window of 512 samples, $SDTW$ is 400 times faster than DTW.*

1. Introduction

Management of data streams has recently emerged as one of the most challenging extensions of database technology. The proliferation of sensor networks as well as the availability of massive amounts of streaming data related to telecommunications traffic monitoring, web-click logs, geophysical measurements and many others, has motivated the investigation of new methods for their modelling, storage, and querying. In particular, continuously monitoring through time the correlation of multiple data streams is of interest in order to detect similar behaviors of stock prices, for video surveillance applications, synchronization of biological signals and, more in general, mining of temporal patterns [Lin et al. 2002, Roddick et al. 2000].

Previous works dealing with the problem of detecting when two or more streams exhibit a high correlation in a certain time interval have tried to extend techniques developed for (*static*) time series to the streaming environment. In particular, Zhu and Shasha [Zhu and Shasha 2002], by adopting a *sliding window* model and the Euclidean distance as a measure of correlation (low distance = high correlation), have been able to monitor in real-time up to 10,000 streams on a PC. However, it is known that for time-varying data a much better accuracy can be obtained if one uses the *Dynamic Time Warping* (DTW) distance [Berndt and Clifford 1994]. Since the DTW can compensate for stretches along the temporal axis, it provides a way to optimally align time series that matches user's intuition of similarity much better than Euclidean distance does, as demonstrated several times (see, e.g., [Ratanamahatana and Keogh 2004] for some recent DTW applications and [Bartolini et al. 2005] for a novel DTW-based approach to shape matching). Further, although not a metric, DTW can be indexed [Keogh 2002], which allows this distance to be applied also in the case of large time series archives.

Unfortunately, when considering streams the benefits of DTW seem to vanish since, unlike Euclidean distance, *it cannot be efficiently updated*. The basic reason is that

the (optimal) alignment one has established at time t is not guaranteed to be still optimal at time $t+1$, thus forcing the DTW to be recomputed from scratch at each time step, with a complexity that, for a sliding window of size n, varies between $O(n)$ and $O(n^2)$, depending on how specific global constraints on valid alignments are set (see Section 2. for details).

Given this unpleasant state of things, in this paper we propose the novel $SDTW$ (*Stream-DTW*) distance measure to be used in place of DTW for the purpose of continuously monitoring the distance of two or more streams. $SDTW$ preserves the ability of DTW to compensate for stretches along the temporal axis, yet, unlike DTW, is efficiently updatable. This precisely means that the work required at each time step can vary from a minimum of $O(1)$ to a maximum of $O(n)$, depending on the global constraints on alignments. In both cases this represents a remarkable $O(n)$ speed-up over DTW computation. Our experiments on five real-world datasets show that $SDTW$ is indeed a very good approximation of DTW, the error never exceeding 10%. We also demonstrate that other approximations of DTW, proposed for the static case, incur much higher errors and have a highly variable behavior over the datasets.

Besides being applicable as a valid alternative to DTW for monitoring purposes, $SDTW$ can also be used as an effective *filtering tool* to speed-up the computation of the result of DTW-based queries in a streaming environment. This holds since we prove (see Theorem 1) that $SDTW$ always lower bounds the DTW. For this querying scenario we provide some preliminary results showing that $SDTW$ is particularly effective in limiting the number of false alarms.

The rest of the paper is organized as follows. In Section 2. we provide the necessary background on the DTW distance. Section 3. highlights the limits of applying the DTW distance in a streaming environment and presents a simple extension of the well-known LB_{Keogh} lower-bounding DTW approximation. In section 4. we introduce the novel $SDTW$ distance measure and prove some of its basic properties. Section 5. provides experimental evidence of the efficiency and the accuracy of $SDTW$, and Section 6. concludes and suggests directions for future research activity.

2. Dynamic Time Warping

We start with some basic definitions related to the static case, i.e., for real-valued time series of finite length.

Let $R \equiv R_1^n$ and $S \equiv S_1^n$ be two time series of length n and let R_i (S_i) be the i-th sample of R (resp. S). A common way to compare two time series is by computing their Euclidean distance (L_2) [Yi and Faloutsos 2000]. For the purpose of this paper, and without loss of generality, we drop the square root from the definition of L_2, thus defining the distance between R and S as:

$$L_2(R_1^n, S_1^n) = \sum_{i=1}^{n} (R_i - S_i)^2 \tag{1}$$

It is important to realize that L_2 (and similar metrics as well, such as the Manhattan (L_1) and the "max" (L_∞) distances) only compares corresponding samples, thus it does not allow for local non-linear *stretches* along the temporal axis.[1] As a consequence,

[1] Observe that the problem of global or uniform stretching is quite different, see [Keogh et al. 2004].

two time series might lead to a high L_2 value even when they are very similar, which can have negative effects on common mining tasks, such as classification and clustering [Chu et al. 2002]. This well-known problem is solved by the *Dynamic Time Warping* (DTW) distance [Berndt and Clifford 1994]. The key idea of DTW is that any point of a series can be (forward and/or backward) aligned with multiple points of the other series that lie in different temporal positions, so as to compensate for temporal shifts.

The definition of DTW is based on the notion of *warping path*. Let d be the $n \times n$ matrix of pairwise squared distances between samples of R and S, $d[i,j] = (R_i - S_j)^2$. A warping path $W = \langle w_1, w_2, \ldots, w_K \rangle$ is a sequence of K ($n \leq K \leq 2n-1$) matrix cells, $w_k = [i_k, j_k]$ ($1 \leq i_k, j_k \leq n$), such that:

boundary conditions: $w_1 = [1,1]$ and $w_K = [n,n]$, i.e., W starts in the lower-left cell and ends in the upper-right cell;

continuity: given $w_{k-1} = [i_{k-1}, j_{k-1}]$ and $w_k = [i_k, j_k]$, then $i_k - i_{k-1} \leq 1$ and $j_k - j_{k-1} \leq 1$. This ensures that the cells of the warping path are adjacent;

monotonicity: given $w_{k-1} = [i_{k-1}, j_{k-1}]$ and $w_k = [i_k, j_k]$, then $i_k - i_{k-1} \geq 0$ and $j_k - j_{k-1} \geq 0$, with at least one strict inequality. This forces W to progress over time.

Any warping path W defines an alignment between R and S and, consequently, a cost to align the two series. The (squared) DTW distance is defined as the minimum of such costs, i.e., the cost of the *optimal warping path*, W_{opt}:

$$DTW(R_1^n, S_1^n) = \min_W \{ \sum_{[i_k,j_k] \in W} d[i_k, j_k] \} = \sum_{[i_k,j_k] \in W_{opt}} d[i_k, j_k] \quad (2)$$

The DTW distance can be recursively computed using an $O(n^2)$ dynamic programming approach that fills the cells of a *cumulative distance* D matrix using the following recurrence relation (see also Figure 1):

$$D[i,j] = d[i,j] + \min\{D[i-1,j-1], D[i-1,j], D[i,j-1]\} \quad (1 \leq i,j \leq n) \quad (3)$$

and then setting $DTW(R_1^n, S_1^n) = D[n,n]$. Note that $D[i,j] = DTW(R_1^i, S_1^j)$, $\forall i,j$, since the DTW distance is also defined when the two series have different lengths [Berndt and Clifford 1994].

In practical applications, warping paths are commonly subject to some *global constraints*, in order to prevent pathological alignments. The most commonly used of such constraints is the Sakoe-Chiba band [Sakoe and Chiba 1978] of width b, that forces warping paths to deviate no more than b steps from the matrix diagonal (see Figure 1 (b)). It is worth noting that, besides reducing the complexity of computing DTW to $O(nb)$, rather surprisingly a band of limited width b leads to better results in classification tasks, even when b is a small fraction (say, $3 \div 4\%$) of n [Ratanamahatana and Keogh 2004].

3. Data Streams: What is the Problem?

Let us now turn to a streaming environment, in which R and S are possibly infinite sequences of samples. We assume that at each time step a new sample for both streams becomes available (i.e., streams are synchronized). We consider the well-known *sliding window* model, according to which, for a sliding window of size n, only the last n points

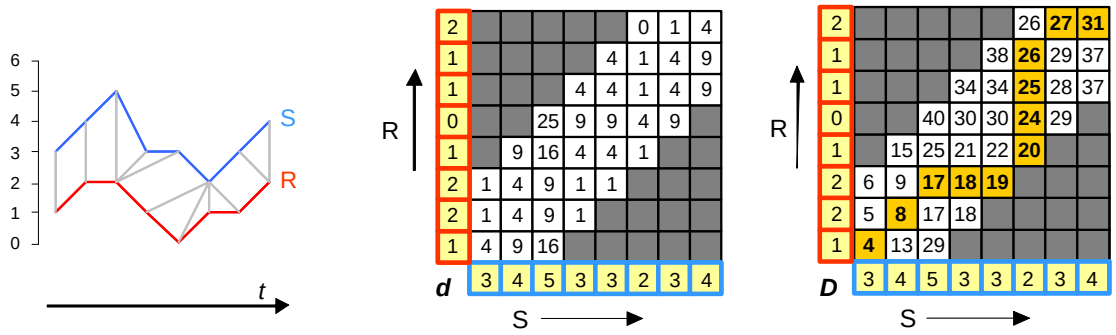


Figure 1. Computing the DTW distance with a Sakoe-Chiba band of width $b = 2$: (a) Optimal alignments; (b) The d matrix of pairwise sample distances; (c) The D cumulative distance matrix; highlighted cells constitute the optimal warping path W_{opt}

of the streams are significative [Babcock et al. 2002, Babu and Widom 2001]. Thus, at time t, the subsequence of stream R falling in the current (active) window is R_{t-n+1}^{t}. At time $t+1$, sample R_{t+1} arrives and R_{t-n+1} expires, thus the new active window consists of R_{t-n+2}^{t+1}. The same is true for stream S.

If one wants to continuously monitor the distance of the streams a näive approach would be to compute, at time $t+1$, the distance between R_{t-n+2}^{t+1} and S_{t-n+2}^{t+1} from scratch, i.e., without considering the computation performed at previous time steps. This might be a challenging task, especially with high sampling rates, large sliding windows, and many (hundreds, thousands) streams to monitor [Zhu and Shasha 2002]. Indeed, these scenarios call for a method that is not only accurate in comparing streams, yet it is also efficient, that is, able to exploit previously performed computation.

As anticipated in the Introduction, the DTW distance is not efficiently updatable, which is also to say that the work to be performed *at each time step* is $O(nb)$. This can be argued from the following observation. Let $W_{opt}(t)$ be the optimal warping path for subsequences R_{t-n+1}^{t} and S_{t-n+1}^{t} and $W_{opt}(t+1)$ be the one for R_{t-n+2}^{t+1} and S_{t-n+2}^{t+1}. Then, in the limit case it could well be the case that $W_{opt}(t)$ and $W_{opt}(t+1)$ share no common cell (alignment) at all, i.e., the optimal alignment established at time t is of no help in finding the optimal alignment at time $t+1$.

Given this unpleasant state of things, we are left with the problem of devising a method for continuous data streams that satisfy both following requirements:

1. It should be a good approximation of DTW. This is because of the success that DTW has already demonstrated for the static case (i.e., time series).
2. It should be fast to update, i.e., its complexity should be at most $O(b)$. Note that this is the best one can achieve with a DTW-like distance measure, since at each new time step $2b+1$ distances between samples have to be necessarily computed. In particular, if b does not vary with n, update complexity should be independent of the sliding window size, n.

Note that the DTW distance trivially satisfies the 1st requirement but not the second, whereas the opposite is true for the Euclidean distance and similar metrics.

3.1. Extending Approximation Techniques Proposed for the Static Case

One possible approach to satisfy both above-stated requirements is to extend to the streaming environment (lower-bounding) approximations of the DTW distance that have been developed for the static case, where they are used to speed-up DTW-based queries over large time series databases [Yi et al. 1998, Kim et al. 2001, Keogh 2002]. In this setting, we are given a (query) time series Q and a search threshold ϵ, and want to retrieve all the time series S in the database for which $DTW(Q,S) \leq \epsilon$ holds. The idea of such methods is to introduce a lower-bounding distance measure, d_{lb}, and to use it to discard all series S such that $d_{lb}(Q,S) > \epsilon$, thus *without* performing the (costly) computation of $DTW(Q,S)$. This works since $d_{lb}(Q,S) > \epsilon$ implies $DTW(Q,S) > \epsilon$ if d_{lb} lower bounds DTW. The DTW computation needs to be executed only for those series that survive the *filtering* step. The effectiveness of this filter & refine approach critically depends on two contrasting factors [Agrawal et al. 1993, Ciaccia and Patella 2002]: d_{lb} should be cheap to compute and, at the same time, it should have a good accuracy, so as to limit the number of *false alarms*, i.e., the number of series for which both $d_{lb}(Q,S) \leq \epsilon$ and $DTW(Q,S) > \epsilon$ hold.

The best-so-far known method to approximate from below the DTW is due to Keogh [Keogh 2002], and consists in constructing an *envelope*, $Env(Q)$, around Q, after that an Euclidean-like distance between $Env(Q)$ and S is computed. More in detail, let $Up(Q)$ and $Low(Q)$ be two time series defined as follows (see Figure 2 (a)):

$$\begin{aligned} Up_i(Q) &= \max\{Q_j | j \in [\max\{1, i-b\}, \min\{n, i+b\}]\} \\ Low_i(Q) &= \min\{Q_j | j \in [\max\{1, i-b\}, \min\{n, i+b\}]\} \end{aligned}$$

Thus, $Up_i(Q)$ ($Low_i(Q)$) is simply the maximum (resp. the minimum) of Q_j values in the interval $[i-b, i+b]$, centered in i and of width $2b+1$. The definition is then completed so as to properly take into account border effects. Consequently, $Env(Q) = (Up(Q), Low(Q))$ encloses all the allowed stretches of Q. Finally, the (squared) LB_{Keogh} distance between

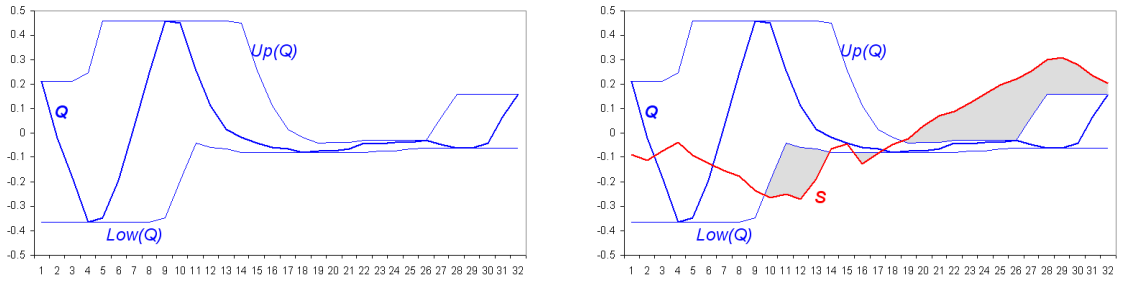


Figure 2. (a) The envelope of Q when $b = 4$; (b) Keogh's lower-bounding distance. In the example: $LB_{Keogh}(Env(Q), S) = 0.416$

$Env(Q)$ and S is defined as:

$$LB_{Keogh}(Env(Q), S) = \sum_{i=1}^{n} \begin{cases} (S_i - Up_i(Q))^2 & \text{if } S_i > Up_i(Q) \\ 0 & \text{if } S_i \in [Low_i(Q), Up_i(Q)] \\ (Low_i(Q) - S_i)^2 & \text{if } Low_i(Q) > S_i \end{cases} \quad (4)$$

and corresponds to the shaded area in Figure 2 (b).

We can easily adapt Keogh's method to deal with streams as follows. Let $LB_{Keogh}(Env(R^t_{t-n+1}), S^t_{t-n+1})$ be the lower-bounding distance between (the envelope of) R^t_{t-n+1} and S^t_{t-n+1}. Since in our scenario the two streams R and S play a symmetric role, we can similarly compute $LB_{Keogh}(Env(S^t_{t-n+1}), R^t_{t-n+1})$ and then, in order to improve accuracy, define the *symmetric* LB_{Keogh} lower-bounding distance measure as (see Figure 3):

$$LB^{symm}_{Keogh}(R^t_{t-n+1}, S^t_{t-n+1}) = \max\{LB_{Keogh}(Env(R^t_{t-n+1}), S^t_{t-n+1}), LB_{Keogh}(Env(S^t_{t-n+1}), R^t_{t-n+1})\} \quad (5)$$

Note that when updating LB^{symm}_{Keogh}, only the first b and the last b values of the envelope can possibly change upon arrival of the new stream samples.

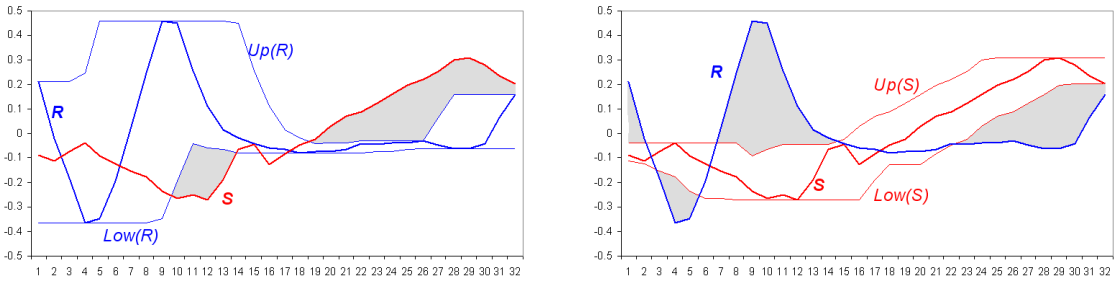


Figure 3. The symmetric LB^{symm}_{Keogh} lower-bounding distance: (a) The envelope of R; (b) The envelope of S. In the example: $LB^{symm}_{Keogh}(R, S) = 1.032$

Depending on the width b of the band and on the specific dataset, the quality of approximation of LB_{Keogh} with respect to DTW is known to wildly vary (see, e.g., [Keogh 2002, Ratanamahatana and Keogh 2004]). Thus, as also our experiments confirm (see Section 5.), using LB_{Keogh} (as well as its symmetric version) in place of DTW for monitoring purposes is not advisable, since it only satisfies the 2nd of our requirements (fast to update) but not the 1st one (good approximation of DTW). Thus, LB^{symm}_{Keogh} could only be considered to be used as a filter for DTW-based queries, as done in the static case.

4. The Stream-DTW Distance

The key to satisfy both requirements of accuracy and efficiency starts from a couple of simple observations. First, in order to obtain a high accuracy, a DTW-like style of computation should be preserved (note that this is not the case with LB^{symm}_{Keogh}). Second, in order to save computational resources, one should realize that the major source of difficulty in updating the DTW distance lies in its *boundary conditions*. To see why this is the case, consider again the optimal warping paths $W_{opt}(t)$ and $W_{opt}(t+1)$. Since $W_{opt}(t+1)$ has necessarily to start from cell $[t-n+2, t-n+2]$, it is evident that, in order to reuse at least part of the computation performed to determine $W_{opt}(t)$, this path has to pass through such cell. However, this is unlikely to be the case.

Clearly, we cannot be confident in ensuring that an optimal warping path passes through a *single* cell, yet we can guarantee that it passes through one of a (suitable chosen) set of cells. This is formalized by the following preliminary definition (see also Figure 4).

Definition 1 (Frontier) *A* frontier F *is any set of cells of the cumulative distance matrix such that for any warping path* W *it is* $W \cap F \neq \emptyset$.
The $\llcorner$*-frontier anchored in cell* $[i, i]$ *is the set of* $2b + 1$ *cells*

$$\llcorner[i, i] = \{[i, i], [i, i + 1], \ldots, [i, i + b], [i + 1, i], \ldots, [i + b, i]\}.$$

The $\urcorner$*-frontier anchored in cell* $[i, i]$ *is the set of* $2b + 1$ *cells*

$$\urcorner[i, i] = \{[i, i], [i, i - 1], \ldots, [i, i - b], [i - 1, i], \ldots, [i - b, i]\}.$$

Note that any warping path (including W_{opt}) has to pass through a frontier.

Now we define two relaxed versions of DTW, which will be used in the definition of our new distance measure.

Definition 2 (Boundary-relaxed DTW) *Given streams* R *and* S*, let* t_s *and* t_e *be two generic time instants. We define:*

- *The* start-relaxed *DTW* ($DTW^{\llcorner}$) *between subsequences* $R_{t_s}^{t_e}$ *and* $S_{t_s}^{t_e}$ *is the value of* $D^{\llcorner}[t_e, t_e]$*, where* $D^{\llcorner}$ *is the* start-relaxed *cumulative distance matrix initialized as follows:*

$$D^{\llcorner}[t_s, t_s + j] = d[t_s, t_s + j]; \quad D^{\llcorner}[t_s + j, t_s] = d[t_s + j, t_s] \qquad (0 \leq j \leq b) \quad (6)$$

  and then completed using the recurrence relation defined in Eq. 3.
- *The* start-end-relaxed *DTW* ($DTW^{\llcorner\urcorner}$) *between subsequences* $R_{t_s}^{t_e}$ *and* $S_{t_s}^{t_e}$ *is:*

$$DTW^{\llcorner\urcorner}(R_{t_s}^{t_e}, S_{t_s}^{t_e}) = \min\{DTW^{\llcorner}(R_{t_s}^{i}, S_{t_s}^{j}) \mid [i, j] \in \urcorner[t_e, t_e]\} \qquad (7)$$

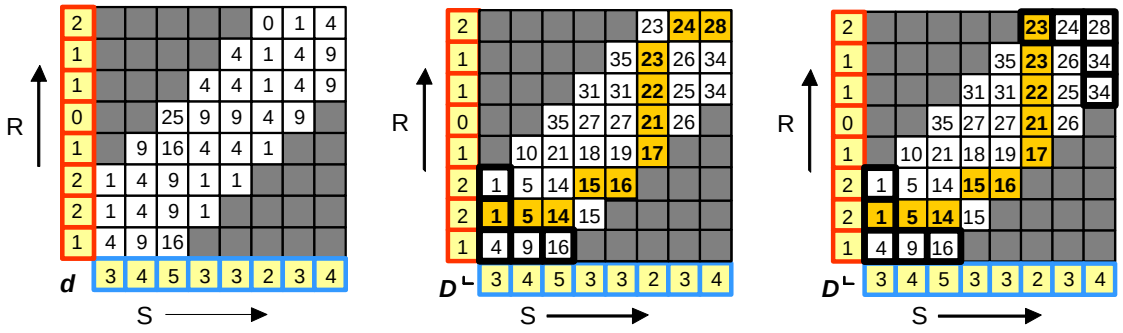


Figure 4. (a) Distance matrix; (b) Start-relaxed DTW**; (c) Start-end-relaxed** DTW

Note that $DTW^{\llcorner}$ computation proceeds as with DTW, yet all the cells in the $\llcorner$-frontier anchored in t_s have as value the distance between the corresponding samples (see Figure 4 (b)). This is to say that warping paths can start from *any* cell in $\llcorner[t_s, t_s]$. When $t_e < t_s$ we conventionally set $DTW^{\llcorner}(R_{t_s}^{t_e}, S_{t_s}^{t_e}) = 0$.

For computing $DTW^{\llcorner\urcorner}(R_{t_s}^{t_e}, S_{t_s}^{t_e})$ one also relaxes the end-boundary condition. In practice one can still use the same start-relaxed $D^{\llcorner}$ matrix used for computing

$DTW^{\llcorner}(R_{t_s}^{t_e}, S_{t_s}^{t_e})$, and then just look at the minimum value on the $\urcorner[t_e, t_e]$ frontier (see Figure 4 (c)).

The basic rationale underlying the computation of the Stream-DTW ($SDTW$) distance is to split, using frontiers, the optimal warping path of the DTW into 2 distinct pieces (one of them possibly null at some time instants): The 1st piece starts by spanning the whole current window of size n, and then, at each time step, progressively reduces; the 2nd piece starts empty and then progressively grows. After exactly n time steps everything starts again. For each of the 2 pieces of W_{opt} the $SDTW$ provides an accurate approximation.

We first present the formal definition of $SDTW$, a detailed explanation of how it works is provided in the Proof of Theorem 1. To stay general, we consider that we want to measure the distance between subsequences $R_{t_s}^{t_e}$ and $S_{t_s}^{t_e}$, where $t_e = kn + i$ for some positive integer k, $0 \leq i < n$, and $t_s = t_e - n + 1 = kn + i - n + 1 = (k-1)n + 1 + i$.

Definition 3 *The Stream-DTW ($SDTW$) distance between subsequences $R_{t_s}^{t_e}$ and $S_{t_s}^{t_e}$ is defined as:*

$$\begin{aligned} SDTW(R_{(k-1)n+1+i}^{kn+i}, S_{(k-1)n+1+i}^{kn+i}) &= DTW^{\llcorner\urcorner}(R_{(k-1)n+1}^{kn}, S_{(k-1)n+1}^{kn}) \\ - \quad & (DTW^{\llcorner}(R_{(k-1)n+1}^{(k-1)n+1+i}, S_{(k-1)n+1}^{(k-1)n+1+i}) - d(R_{(k-1)n+1+i}, S_{(k-1)n+1+i})) \\ + \quad & DTW^{\llcorner}(R_{kn+1}^{kn+i}, S_{kn+1}^{kn+i}) \end{aligned} \tag{8}$$

Theorem 1 (Lower bound) *The $SDTW$ distance is a lower bound of DTW.*

Proof. For the sake of conciseness, denote with α, β, γ and δ the 4 terms in the right-hand side of Eq. 8, so that we have to show that $\alpha - (\beta - \gamma) + \delta$ is a lower bound of DTW (see also Figure 5).

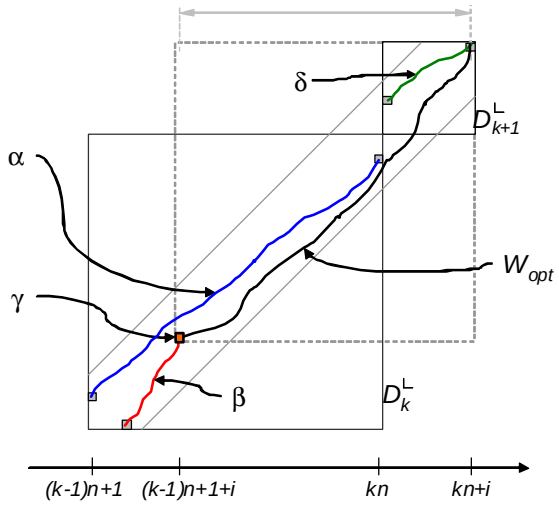


Figure 5. How $SDTW$ works ($SDTW = \alpha - (\beta - \gamma) + \delta$)

Let W_{opt} be the optimal warping path for aligning $R_{(k-1)n+1+i}^{kn+i}$ and $S_{(k-1)n+1+i}^{kn+i}$, and consider the frontiers $\urcorner[kn, kn]$ and $\llcorner[kn+1, kn+1]$. Consider the 1st part of W_{opt}, call it $W_{opt,k}$, that ends in a cell of $\urcorner[kn, kn]$, and the 2nd part of W_{opt}, call it $W_{opt,k+1}$,

that starts from a cell in $\llcorner[kn+1, kn+1]$. We claim that $\alpha - (\beta - \gamma)$ lower bounds the DTW contribution, call it DTW_k, corresponding to $W_{opt,k}$ and that δ lower bounds the component, call it DTW_{k+1}, corresponding to $W_{opt,k+1}$. Since the DTW distance between the two subsequences is $\geq DTW_k + DTW_{k+1}$ this will prove the result.
$[\alpha - (\beta - \gamma) \leq DTW_k]$. Consider the start-relaxed path, call it W_β, corresponding to β and ending in cell $[t_s, t_s]$, and the path $W_{opt,k}$, which shares cell $[t_s, t_s]$ with W_β. Counting just once the contribution of cell $[t_s, t_s]$, i.e., γ, we end up with a total cost given by $\beta - \gamma + DTW_k$ for going from $\llcorner[(k-1)n+1, (k-1)n+1]$ to $\urcorner[kn, kn]$. From the definition of $DTW^{\llcorner\urcorner}$, this cannot be less than α, which proves the assert.
$[\delta \leq DTW_{k+1}]$. Immediate from the definition of start-relaxed DTW. $\square$

Above proof shows how we can approximate from below the DTW by splitting the optimal warping path into 2 pieces. The two frontiers we use to this purpose are by no means the only possible ones; in particular, as Figure 5 shows, a part of W_{opt} could traverse some cells, after leaving $\urcorner[kn, kn]$ and before entering $\llcorner[kn+1, kn+1]$, that $SDTW$ does not consider at all. We can prove that our arguments are still applicable should we replace $\llcorner[kn+1, kn+1]$ with the $\urcorner[kn+1, kn+1]$ frontier. We do not enter into further details here.

It is evident that, unlike simpler approximations such as LB_{Keogh}, the computation of $SDTW$ shares with DTW the need of computing an optimal warping path, even if on a start-relaxed cumulative distance matrix. This alone suggests that the very first instant we compute the $SDTW$ we have to pay a cost of $O(nb)$. However, as the following theorem shows, $SDTW$ is amenable to be efficiently updatable.

Theorem 2 (Update Complexity) *The $SDTW$ distance can be updated in time $O(b)$ at any time step.*

Proof. Denote with α', β', γ' and δ' the new values, at time step $t_{e'} = t_e + 1 = kn + i + 1$, of the 4 terms in Eq. 8. Let $D^{\llcorner}_k$ be the start-relaxed cumulative distance matrix for time interval $[(k-1)n + 1 : kn]$, and $D^{\llcorner}_{k+1}$ be the one for the interval $[kn + 1 : (k+1)n]$. Let d_k and d_{k+1} be the corresponding matrices storing distances between samples of R and S.

The steps needed to update the value of $SDTW$ include the computation of distance values for the two new samples of R and S, and the extension of matrix $D^{\llcorner}_{k+1}$ up to frontier $\urcorner[t_{e'}, t_{e'}] = \urcorner[t_e + 1, t_e + 1]$. Both steps require $O(b)$ time.

Consider now the case when $t_{e'} = kn + i + 1$, with $0 \leq i < n - 1$. We have:

- $\alpha' = \alpha$, since this term does not depend on i.
- By definition of start-relaxed DTW and of $D^{\llcorner}_k$, it is $\beta' = D^{\llcorner}_k[(k-1)n+1+i+1 : (k-1)n+1+i+1]$, i.e., computing β' costs $O(1)$. The same is clearly true for $\gamma' = d(R_{(k-1)n+1+i+1}, S_{(k-1)n+1+i+1})$.
- Finally, $\delta' = D^{\llcorner}_{k+1}[kn + i + 1 : kn + i + 1]$, again with cost $O(1)$.

When $i = n - 1$, it is $t_{e'} = kn + (n-1) + 1 = (k+1)n$ and we have $\beta' = \gamma'$ and $\delta' = 0$ (by definition of start-relaxed DTW). The new $SDTW$ value reduces to $\alpha' = DTW^{\llcorner\urcorner}(R^{(k+1)n}_{kn+1}, S^{(k+1)n}_{kn+1})$, which, given matrix $D^{\llcorner}_{k+1}$, can be computed in $O(2b+1) = O(b)$ time. $\square$

Finally, the following result shows that the memory required by $SDTW$ is of the same order of that needed to continuously update the Euclidean distance.

Theorem 3 (Space Complexity) *The $SDTW$ distance can be computed using $O(n+b) = O(n)$ space.*

Sketch of proof. Refer to Figure 5 and observe that, starting from time $t_e = kn + 1$, all information in matrices d_z and $D^{\llcorner}_z$, with $1 \leq z < k$, is not needed anymore to determine future $SDTW$ values. This immediately bounds the space required to $O(nb)$. In other terms, as Figure 5 suggests, at each time step at most 2 distance matrices and 2 cumulative distance matrices are needed.

Consider now the proof of Theorem 2. To reduce the space to $O(n + b)$ it is sufficient to observe that, when $t_e \in [kn + 1 : (k + 1)n]$, all one needs to know about the d_k and $D^{\llcorner}_k$ matrices is just the difference of corresponding diagonal elements (i.e., the $(\beta - \gamma)$ terms) and the (single) α value. This accounts for the n term in $O(n+b)$. Finally, the b term is due to the need of storing the last $\urcorner$-frontier of matrix $D^{\llcorner}_{k+1}$, which is used to update the δ term. □

5. Experimental Results

In this section we present experimental results aiming to evaluate the actual performance of $SDTW$, both in terms of efficiency and of quality of approximation. For reference purpose, we contrast $SDTW$ to the LB^{symm}_{Keogh} method presented in Section 3.1.. Our experiments are performed on a 1.60GHz Intel Pentium 4 CPU with 512 MB of main memory and running Windows 2000 OS. We use five different real-world datasets, obtained from the UCR archive [Keogh and Folias 2002] (see Figure 6 for samples from such datasets):

- *Random walk*: classical random walk data, generated according to the equation $S_i = S_{i-1} + rand(-0.5, 0.5)$;
- *Burstin*: this dataset includes 12 hours of data from a small region of the sky, where Gamma Ray bursts were reported during that period;
- *Fluid dynamics*: this stream represents one of the signals collected at a frequency of 500Hz from boundary layer experiments in the fluid dynamics research domain;
- *Network*: this dataset describe the Round Trip Time (RTT) delay of a sequence of packets sent from University of California at Riverside to Carnegie Mellon University;
- *EEG*: 128Hz-electroencephalographic data acquired at the Department of Physiology (University of Bologna).

For each of the above datasets we simulate acquisition of new samples by reading from memory a new data sample at fixed time intervals (sampling rate). In order to generate multiple streams of a same kind, we partition each dataset into non-overlapping parts, and then take each of such parts as a different stream.

Our first experiment aims to show the actual speed-up obtainable from $SDTW$ with respect to DTW. In Figure 7 we show how many streams we can monitor in real time for the methods under analysis when the sliding window includes $n = 128$ samples. More precisely, for a given number of pairs of streams to be compared to each other, we vary the sampling rate and plot the value beyond which we cannot report results before

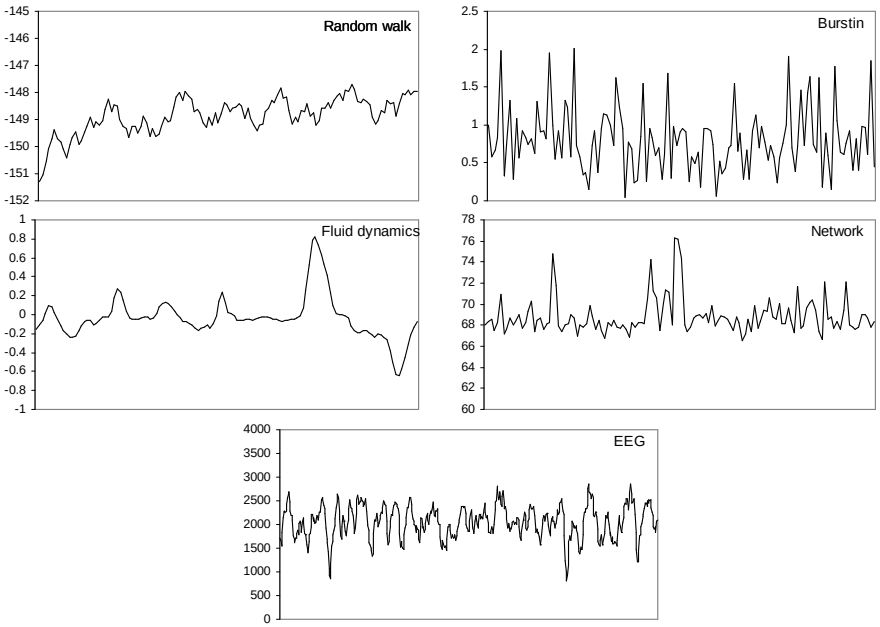


Figure 6. Samples from the five datasets used in the experiments

the next stream samples have to be read. Thus, working, say, at 1000 Hertz means that all the distance values between the pairs of streams are reported no later than 1 millisecond.

It is evident that $SDTW$ outperforms DTW by up to two orders of magnitude. For instance, at 100 Hertz $SDTW$ can monitor up to about 100 pairs of streams, whereas DTW can only handle 1 pair. The LB_{Keogh}^{symm} method has an intermediate behavior, since it pays the overhead of updating the envelopes of all streams to be compared.

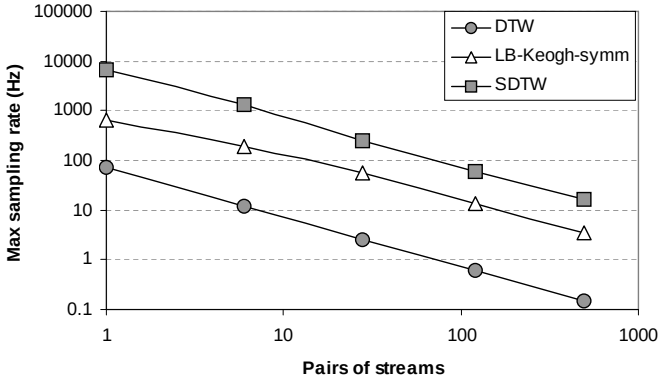
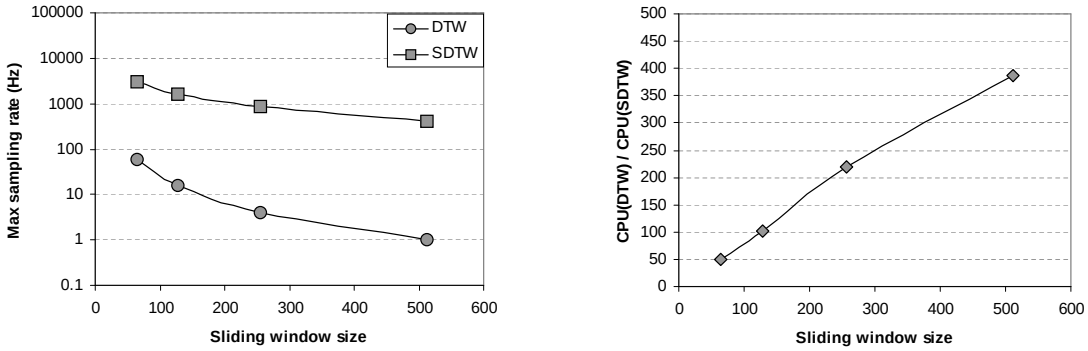


Figure 7. Maximum number of pairs of streams that can be monitored with DTW, LB_{Keogh}^{symm} and $SDTW$. Sliding window size $n = 128$. Sakoe-Chiba band size $b = 8$. Results are averaged over the 5 datasets in Figure 6

Figure 8 (a) shows that the gain of $SDTW$ with respect to DTW actually improves with the sliding window size, and Figure 8 (b) clearly makes evidence that the

ratio $CPU(DTW)/CPU(SDTW)$, where $CPU()$ measures the CPU cost of a method, indeed grows as $O(n)$, as expected.



(a) (b)

Figure 8. (a) Maximum sampling rate vs. length of sliding window, n; (b) Ratio of average CPU costs, $CPU(DTW)/CPU(SDTW)$. 6 pairs of streams. Sakoe-Chiba band size $b = n * 6.25\%$. Results are averaged over the 5 datasets in Figure 6

Figure 9 analyzes the accuracy (or *tightness*) of $SDTW$ and LB^{symm}_{Keogh} with respect to DTW. The tightness is measured as the average of $SDTW/DTW$ and LB^{symm}_{Keogh}/DTW ratios over 496 pairs of streams. As it can be seen, the tightness of $SDTW$ is very stable over the different datasets, and is always at least 90%. This is not the case with LB^{symm}_{Keogh}, whose tightness never exceeds about 61%, and is as low as less than 8% for the *Burstin* dataset.

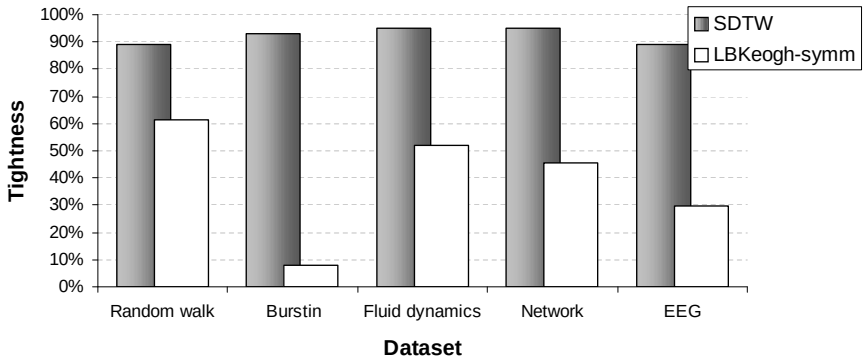


Figure 9. The tightness of $SDTW$ and LB^{symm}_{Keogh} with respect to DTW; Sliding window size $n = 128$. Sakoe-Chiba band size $b = 8$

Our last experiment aims to shed more light on the effects of the tightness in the case one is interested in performing DTW-based queries on streams. More precisely, we consider the number of false alarms yielded by $SDTW$ and LB^{symm}_{Keogh} when the task is to determine whether the DTW distance falls below a given threshold value ϵ. Remind that, according to Theorem 1, $SDTW$ can be safely used to this purpose since it is a lower bound of DTW.

Clearly, as explained in Section 3.1., a high number of false alarms leads to too many unnecessary DTW computations, thus slowing-down the overall querying process. Figure 10 reports the percentage of false alarms for the different datasets, computed as the ratio of the number of times the filtering distance (either $SDTW$ or LB_{Keogh}^{symm}) falls under ϵ to the total number of comparison between stream subsequences. The experimental setting is the same as in Figure 9 (496 pairs of streams, $n = 128$ and $b = 8$) and ϵ is chosen so as to have a 1% selectivity, that is, only in 1% of cases the DTW distance is under the threshold.

It is evident that the good tightness of $SDTW$ pays off, in that false alarms are never more than 1.5% (0.85% averaged over all datasets). For the *Fluid dynamics* dataset no false alarm at all occurs. This and *Random walk* are the only datasets for which LB_{Keogh}^{symm} yields a limited number of false alarms (4.5% and 6%, respectively), whereas in the other cases the effectiveness of the filter step is seriously compromised. Overall, the false alarms percentage of LB_{Keogh}^{symm} is 41.13%, peaking to 99% for the *Burstin* dataset.

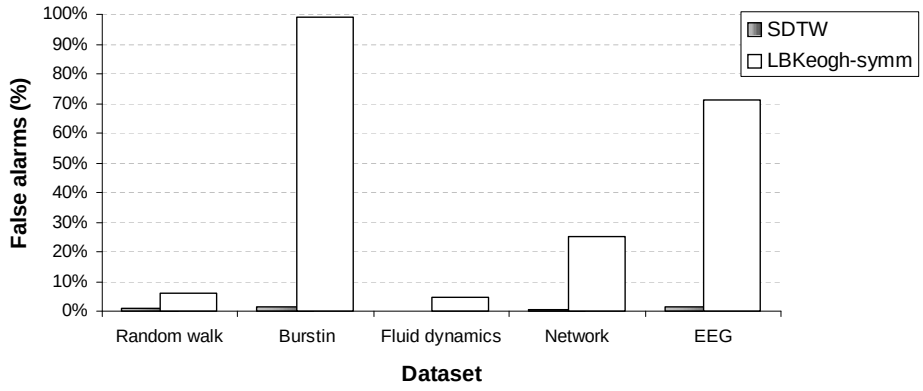


Figure 10. Percentage of false alarms

6. Conclusions

In this paper we have proposed a novel method, called $SDTW$ (*Stream-DTW*) to compare data streams that, unlike the well-known Dynamic Time Warping (DTW) distance, can be efficiently updated when a window slides over the streams. Our experimental results confirm the theoretical analysis and show that the improvement in efficiency of $SDTW$ over DTW linearly grows with the size of the sliding window, thus making it possible to compute a DTW-like distance measure even on (very) large sliding windows. We have also shown that extensions of methods developed for the static case, i.e., when time series are known in advance, and there used to speed-up the evaluation of DTW-based queries cannot cope with $SDTW$ in terms of accuracy.

Our results open a series of interesting research directions and possibilities. An efficiently updatable DTW-like distance measure makes it possible to perform on streams several analysis developed only for the static case, such as those related to mining tasks. For instance, $SDTW$ could be compared to the Euclidean distance in clustering and classification tasks, as already done with static time series. The extension of $SDTW$

to non-synchronized data streams, where the arrival frequency of new data varies for each stream as well as within a single stream is also an interesting research issue.

In this paper we have mainly focused on the problem of monitoring the distance of streams and proposed $SDTW$ as an *alternative* to DTW. We have also suggested that, due to its lower-bounding property, $SDTW$ can also be used as a filtering step if one wants to perform DTW-based queries on streams. Although for this scenario we have shown that $SDTW$ will lead to very few false alarms, one should be aware that bursts of stream subsequences for which the DTW distance needs to be computed are always possible. Therefore, since this will necessarily introduce some delay in the evaluation process, it is an interesting problem to precisely understand how the distribution of such delays will vary with the number and the characteristics of the streams.

Finally, a major issue that remains to be addressed concerns the definition of a method able to compute a DTW-like distance on streams that need to be *normalized* on-line. Indeed, working with unnormalized data may have negative effects on the significance of any similarity measure (including DTW), as demonstrated for static time series [Keogh and Kasetty 2003]. Extending our method to normalized data does not seem to be a trivial task. Intuitively, the problem with normalization is that now a data sample no longer has a fixed value, since at each time step the normalization constants (mean and standard deviation) may change. Reusing computations performed at previous time steps then becomes a major challenge.

References

Agrawal, R., Faloutsos, C., and Swami, A. (1993). Efficient Similarity Search in Sequence Databases. In *Proceedings of the 4th International Conference on Foundations of Data Organizations and Algorithms (FODO'93)*, pages 69–84, Chicago, IL, USA.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and Issues in Data Stream Systems. In *Proceedings of the 21st Symposium on Principles of Database Systems (PODS'02)*, pages 1–16, Madison, WI, USA.

Babu, S. and Widom, J. (2001). Continuous Queries over Data Streams. *SIGMOD Record*, 30(3):109–120.

Bartolini, I., Ciaccia, P., and Patella, M. (2005). WARP: Accurate Retrieval of Shapes Using Phase and Time Warping Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(1):142–147.

Berndt, D. J. and Clifford, J. (1994). Using Dynamic Time Warping to Find Patterns in Time Series. In *AAAI 1994 Workshop on Knowledge Discovery in Databases*, pages 359–370, Seattle, WA, USA.

Chu, S., Keogh, E. J., Hart, D., and Pazzani, M. (2002). Iterative Deepening Dynamic Time Warping for Time Series. In *Proceedings of the Second SIAM International Conference on Data Mining (SDM'02)*, Arlington, VA, USA.

Ciaccia, P. and Patella, M. (2002). Searching in Metric Spaces with User-defined and Approximate Distances. *ACM Transactions on Database Systems*, 27(4):398–437.

Keogh, E. (2002). Exact Indexing of Dynamic Time Warping. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 406–417, Hong Kong, China.

Keogh, E. and Kasetty, S. (2003). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371.

Keogh, E., Palpanas, T., Zordan, V. B., Gunopulos, D., and Cardie, M. (2004). Indexing Large Human-Motion Databases. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04)*, pages 780–791, Toronto, Canada.

Keogh, E. J. and Folias, T. (2002). The UCR Time Series Data Mining Archive. Available at URL `http://www.cs.ucr.edu/~eamonn/TSDMA/index.html`.

Kim, S.-W., Park, S., and Chu, W. W. (2001). An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 607–614, Heidelberg, Germany.

Lin, W., Orgun, M. A., and Williams, G. J. (2002). An Overview of Temporal Data Mining. In *Proceedings of the 1st Australian Data Mining Workshop (ADM'02)*, pages 83–90, Canberra, Australia.

Ratanamahatana, C. A. and Keogh, E. J. (2004). Making Time-Series Classification More Accurate Using Learned Constraints. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, Lake Buena Vista, FL, USA.

Roddick, J. F., Hornsby, K., and Spiliopoulou, M. (2000). An Updated Bibliography of Temporal, Spatial, and Spatio-temporal Data Mining Research. In *Temporal, Spatial, and Spatio-Temporal Data Mining: First International Workshop (TSDM 2000)*, pages 147–164, Lyon, France.

Sakoe, H. and Chiba, S. (1978). A Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

Yi, B.-K. and Faloutsos, C. (2000). Fast Time Sequence Indexing for Arbitrary Lp Norms. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 385–394, Cairo, Egypt.

Yi, B.-K., Jagadish, H. V., and Faloutsos, C. (1998). Efficient Retrieval of Similar Time Sequences Under Time Warping. In *Proceedings of the 14th International Conference on Data Engineering (ICDE'98)*, pages 201–208, Orlando, FL, USA.

Zhu, Y. and Shasha, D. (2002). StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 358–369, Hong Kong, China.