FTW: Fast Similarity Search under the Time Warping Distance

Yasushi Sakurai*
NTT Cyber Space Laboratories
sakurai.yasushi@lab.ntt.co.jp

Masatoshi Yoshikawa
Nagoya University
yosikawa@itc.nagoya-u.ac.jp

Christos Faloutsos†
Carnegie Mellon University
christos@cs.cmu.edu

ABSTRACT

Time-series data naturally arise in countless domains, such as meteorology, astrophysics, geology, multimedia, and economics. Similarity search is very popular, and DTW (Dynamic Time Warping) is one of the two prevailing distance measures. Although DTW incurs a heavy computation cost, it provides scaling along the time axis. In this paper, we propose FTW (Fast search method for dynamic Time Warping), which guarantees no false dismissals in similarity query processing. FTW efficiently prunes a significant number of the search candidates, which leads to a direct reduction in the search cost. Experiments on real and synthetic sequence data sets reveal that FTW is significantly faster than the best existing method, up to 222 times.

1. INTRODUCTION

Time-series data naturally occur in many application domains, such as computational biology, meteorology, astrophysics, geology, multimedia and economics. Even though the databases generated by the corresponding applications continue to grow in size, a common demand is to find similarities between time-series data sequences. Moreover, these applications require a sequence-matching mechanism that is robust against noise while providing scaling of the time axis of the sequences.

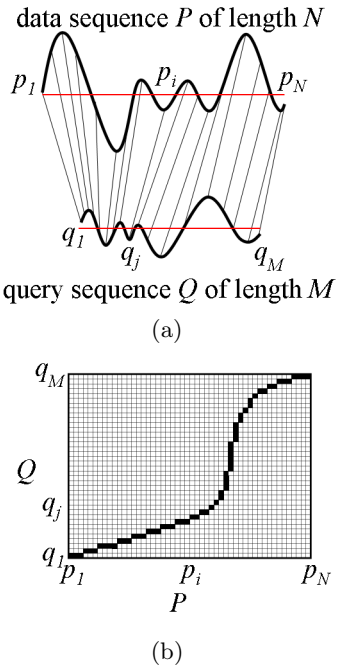


Figure 1: Illustration of DTW. (a) The alignment of measurements by DTW for measuring the distance between two sequences. (b) DTW obtains a mapping between the sequences. The black squares denote the optimum warping path.

Retrieving long sequences is very expensive given the large data sets involved, and various indexing and searching methods have been proposed to reduce this cost. Most of the earlier works on high-speed sequence matching are based on the Euclidean distance function. Since the Euclidean distance function treats sequence elements independently, it cannot be used to calculate the distance between sequences whose lengths and/or sampling rates are different. Furthermore, it can be sensitive to outliers [2]. Recent applications have adopted Dynamic Time Warping (DTW) [5, 22] to overcome these problems [13, 21, 19, 12]. DTW is a transformation that allows sequences to be stretched along the time axis to minimize the distance between the sequences. The distance of DTW is calculated by dynamic programming (See Figure 1 for a drawing and Section 3.1 for the exact definition). The matrix in Figure 1(b) is fundamental for DTW, and we

*Part of this work was done while this author was visiting Carnegie Mellon University.

†This material is based upon work supported by the National Science Foundation under Grants No. IIS-0083148, IIS-0113089, IIS-0209107 IIS-0205224 INT-0318547 SENSOR-0329549 EF-0331657IIS-0326322 CNS-0433540 by the Pennsylvania Infrastructure Technology Alliance (PITA) Grant No. 22-901-0001. Additional funding was provided by Intel and Northrop-Grumman Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

PODS 2005, June 13-15, 2005, Baltimore, Maryland.

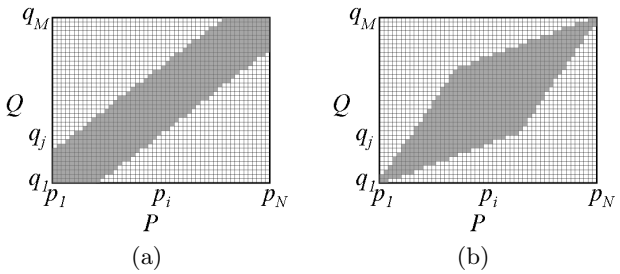


Figure 2: Illustration of global constraints. Global constraints limit the warping scope. The gray area corresponds to the warping scope. (a) Sakoe-Chiba Band. (b) Itakura Parallelogram.

shall refer to it as the *time warping matrix*. The warping path is the set of grid cells in the time warping matrix, which represents an alignment between the sequences. Although DTW incurs a heavy computation cost, it is more robust against noise and provides scaling along the time axis. This ability allows DTW to identify similarities far more accurately and so enhances the functionality of the applications that use it.

The ideal method for DTW should fulfill the following requirements:

1. *Fast*: The exact DTW is quadratic, and prohibitive for long sequences.
2. *No false dismissals*: A search method that returns the qualifying sequences without any omissions is required. It should achieve a high level of search performance even though it ensures no false dismissals.
3. *No restriction on the sequence lengths*: The method should handle any sequence, even data sequences of different lengths and/or data sequences with lengths different from that of the query sequence.
4. *Support for any, as well as for no restriction on warping scope*: The method in [14] is fast, because it cleverly exploits global constraints [22] that appear in dynamic programming (See Figure 2). We would like to have a method that can exploit the restrictions on the warping scope, when the user so desires; but we also want a method that will be fast for the plain DTW, that is, even when the user specifies *no* restrictions warping scope.

Our method, described below, bears all these characteristics, while none of the existing methods can claim the same.

The problem we address in this paper is the following:

PROBLEM 1. *Given S time-series data sequences of unequal lengths $\{P_1, P_2, \ldots, P_S\}$, a query sequence Q, an integer k, and optionally a warping scope W, find the k-nearest neighbors of Q from the data sequence set by using DTW with W.*

The warping scope is the area that the warping path is allowed to visit in the time warping matrix. It can be limited by global constraints as shown in Figure 2. Some constraints are discussed in [22] and [14]. Note that W can be the unrestricted warping scope as well as any restricted warping scope in the problem definition of this paper.

In this paper, we propose FTW (Fast search method for dynamic Time Warping). To obtain the exact time warping distance, we have to compute distances for all possible warping paths; this incurs a high computation cost. In order to reduce the computation time, (1) we propose a new lower bounding measure that approximates the time warping distance, (2) we exclude the warping paths that will not yield fruitful search results by using a new algorithm for dynamic programming, and (3) the search algorithm gradually enhances the accuracy of the distance approximations.

We carried out experiments on real and synthetic sequence data sets. FTW pruned a significant number of data sequences at low computation cost, thus reducing the total search cost. In fact, it is significantly faster than the best existing method, outperforming it by at least *one order of magnitude*, and occasionally up to 222 times. Moreover, the superiority of FTW grows as data size and/or sequence length increases. This tendency makes it more attractive for large and long sequence databases.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes our proposed method, FTW. We show how the approximate distance can be computed, and then introduce our data structure and search algorithm. Section 4 reviews the results of the experiments, which clearly show the effectiveness of FTW. Section 5 is a brief conclusion.

2. RELATED WORK

Agrawal et al. first proposed an approach for similarity sequence matching [1]. Their method extracts feature vectors from sequences, and indexes them using R*-trees. Only a small number of features are extracted, since most multidimensional index structures cannot provide high enough performance for high-dimensional data because of the dimensionality curse problem [4, 23]. Their work focus on whole sequence matching. This was generalized to allow subsequence matching [8, 18].

Keogh et al. presented an indexing method by using the Adaptive Piecewise Constant Approximation (APCA) [15]. APCA is a dimensionality reduction technique for sequence matching based on the Euclidean distance. This technique uses constant-value segments to approximate sequences. While many dimensionality reduction techniques have been proposed (e.g. DFT [20, 10], Discrete Wavelet Transform [24] and Karhunen-Loeve Transform [9]), APCA gives especially high approximation quality.

Recent applications require DTW for calculating the similarity of sequences [13, 21, 19, 12]. To reduce the matching cost, many sequence-matching techniques based on dynamic programming have been proposed, especially in speech recognition [22] and bioinformatics [19]. These techniques trade off

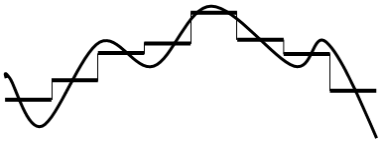


Figure 3: Example of PAA representation for a sequence. In this case, the sequence is reduced to 8 dimensions. Each equal-sized segment of PAA is the average of the sequence during the time period of the segment.

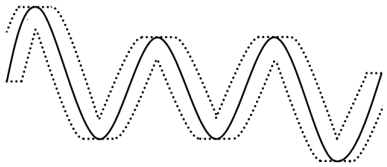


Figure 4: Example of a sequence envelope. The envelope consists of lower and upper bounds that totally enclose the sequence.

sequence retrieval speed against precision. They must visit all data sequences of length N, and the time complexity is still basically $O(N^2)$.

Chu et al. [6] proposed a search method based on distance approximation, which varies its accuracy during the course of query processing. Although this method is efficient, it does not guarantee no false dismissals.

Yi et al. proposed a lower bounding measure for DTW [26]. The distance from a query sequence to each data sequence is evaluated by using a lower bounding measure, after which a candidate set is constructed. The current data sequence for each candidate is visited, and the exact distance between the data sequence and the query sequence is calculated. The lower bounding measure is the sum of the squared differences between the maximum of the query sequence and elements in the data sequence that are greater than that maximum. This lower bounding measure also employs elements in the data sequence that are smaller than the minimum of the query sequence.

Kim et al. introduced a lower bounding measure employing 4-dimensional vectors [17] that represent the first, last, minimum, and maximum sequence elements. The vectors can be readily indexed using any spatial access method. The lower bounding measures proposed in [26] and [17] guarantee no false dismissal. However, the results provided in [14] indicate that these approximations are coarse. The number of exact distance calculations for the search is large, thus leading to high search costs.

Keogh [14] proposed a search method based on global constraints that appear in dynamic programming. Global constraints (e.g. the Sakoe-Chiba Band and the Itakura Parallelogram [22]) limit the scope of the warping path. Zhu et al.'s search method [27] is also based on global constraints and represents an improvement over the one of [14]. The search methods of [14] and [27] compute the envelope of the query sequence from the scope of warping paths; they then derive the PAA (Piecewise Aggregate Approximation) [16, 25] of the envelope (See Figures 3 and 4). The lower bounding distance between each data sequence and the query sequence is defined as the Euclidean distance between the PAA of the envelope and the MBR (Minimum Bounding Rectangle) of the data sequence. Although these search methods are efficient for warping paths with narrow scope, their search performance deteriorates as the warping scope becomes wider. Since the optimum scope depends on the application and data set, search methods that give high search performance, even for wide scopes, are required.

3. PROPOSED METHOD

We propose FTW (Fast search method for dynamic Time Warping), which guarantees no false dismissals. The examples provided in this section assume that there is *no restriction* on the warping scope; recall that is the most expensive setting, that most competing methods can not handle. Note that FTW can also improve search performance when the warping scope is limited by global constraints. Unlike previous works [14, 27], FTW can handle any sequence, even data sequences of different lengths and/or data sequences with lengths different from that of the query sequence.

Range queries and k-nearest neighbor queries are essentially important for practical applications. Although we mainly focus on k-nearest neighbor queries in this paper, FTW can efficiently support queries of both types.

3.1 Preliminaries

Dynamic Time Warping (DTW) is a transformation that temporally warps sequences with the goal being to minimize the distance between the sequences. Consider two sequences $P = \{p_1, p_2, \ldots, p_N\}$ of length N and $Q=\{q_1, q_2, \ldots, q_M\}$ of length M. Their time warping distance $D_{dtw}(P, Q)$ is defined as:

$$
\begin{aligned}
& D_{dtw}(P,Q) = f(N,M) \\
& f(i,j) = \|p_i - q_j\| + \min \begin{cases} f(i,j-1) \\ f(i-1,j) \\ f(i-1,j-1) \end{cases} \\
& f(0,0) = 0, \quad f(i,0) = f(0,j) = \infty \\
& (i = 1, \ldots, N; j = 1, \ldots, M)
\end{aligned} \tag{1}
$$

where $\|p_i - q_j\| = (p_i - q_j)^2$ is the distance between two numerical values. Notice that any other choice (say, absolute difference: $\|p_i - q_j\| = |p_i - q_j|$) would be fine; our upcoming algorithms are completely independent of such choices. The time warping distance between two sequences is obtained by matching each sequence element of P to an element of Q in increasing time order. Since the time warping distance is obtained by using a dynamic programming algorithm whose complexity is $O(NM)$, it can be slow, especially for long sequences.

3.2 Sketch of FTW

Our solution is based on three major ideas, described below. First we give the intuition behind each of them, and in the three subsections we describe each of them in detail.

Symbol	Definition
d_{cb}	the current k-th nearest neighbor distance (i.e., the current best)
P	a data sequence of length N
p_i	the i-th element of P $(i = 0, \ldots, N)$
Q	a query sequence of length M
q_i	the i-th element of Q
p_i^A	the i-th approximate segment of P $(i = 0, \ldots, n)$
P^A	the coarse version of P (P^A=$\{p_1^A, \ldots, p_n^A\}$)
p_i^R	the segment range of p_i^A
p_i^L, p_i^U	the lower and upper bounds of p_i^R
p_i^T	the time interval of p_i^A

Figure 5: Symbols and definitions.

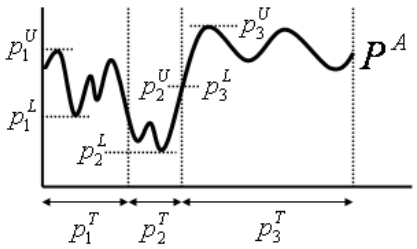


Figure 6: Illustration of sequence approximation. The coarse version P^A of a sequence P has three segments. The i-th approximate segment p_i^A is represented by the time interval p_i^T and the segment range $p_i^R = (p_i^L : p_i^U)$.

LBS: New Lower Bounding Distance Measure

As described in Section 3.1, the complexity of DTW is $O(NM)$, where N and M are sequence lengths. Since computing the exact DWT distance is expensive, especially for long sequences, it is beneficial to use approximations in query processing.

We operate on coarse representations of the sequences, which is obtained by segmentation. Figure 6 shows an example of the representation, called approximate segments. In this figure, we represent the coarse version of a sequence with three segments, each of which consists of its segment ranges and time intervals.

We use the coarse version of sequences to estimate the time warping distance. Figure 7(a-1) illustrates the approximate segments of P and Q, P^A and Q^A. We compute the lower bounding distance from P^A and Q^A by using a dynamic programming approach as shown in Figure 7(a-2). Our algorithms work for segments of arbitrary, even unequal sizes; but for exposition purposes, let's assume that all segments have the same size t. In that case, we achieve $O(\frac{N}{t}\frac{M}{t})$ time. Instead of computing the exact time warping distance for all sequences in the dataset, which needs $O(NM)$ time, we use the new lower bounding distance measure, LBS (Lower Bounding distance measure with Segmentation), and prune a significant number of sequences with $O(\frac{N}{t}\frac{M}{t})$ time.

EarlyStopping: New Algorithm for Dynamic Programming

Query processing maintains the list of candidate k-nearest neighbors before reporting the final k-nearest neighbors. The current k-th nearest neighbor distance, d_{cb} (i.e., the current best), means the exact distance of the best k candidates so far. When a similar sequence, which gives an exact distance less than d_{cb}, is found, the list of candidate k-nearest neighbors has to be updated; this makes d_{cb} smaller. d_{cb} keeps decreasing or remains unchanged; it never increases.

The second idea is to exclude the warping paths that will not yield fruitful search results by using d_{cb}. In other words, we exclude the warping paths by using the exact distance between the query sequence and the candidate k-th nearest neighbor sequence. This makes the distance computation more efficient as d_{cb} becomes smaller. Even if a given warping scope is not restricted (i.e. a global constraint is not used), we can dynamically reduce the warping scope by excluding warping paths that give distances exceeding d_{cb}. That is, d_{cb} serves as the threshold for reducing the warping scope.

Now let us use Figure 7(a-2) again to give an example of this idea. We assume a coarse representation of the time sequences. Let $g(i, j)$ be the lower bounding distance for the grid cell (i, j) (the exact definition of the distance $g(i, j)$ will be given in Equation 2). In the figure, if $g(1, 2) > d_{cb}$, we do not need to compute $g(1, 3)$ since $g(1, 3) \geq g(1, 2)$ always holds. Similarly, we can exclude $g(4, 1)$ if $g(3, 1) > d_{cb}$. The gray area represents the reduced warping scope. We can safely skip the distance computation for a warping path which goes through one or more white areas. We dynamically compute the time warping distance by using d_{cb}. As well as for the approximate time warping distance calculation, this idea can be used for the exact time warping distance calculation (See Figure 7(c)).

"Refinement": New Search Algorithm for DTW

Instead of operating on approximate segments of a single granularity, we propose to use multiple granularities, trying to balance the trade-off between accuracy and comparison speed. As the approximate segment becomes more accurate, the approximate distance often increases, but the computation cost also grows. There is a tradeoff between the approximation distance and the computation cost. Accordingly, we gradually increase the granularity of sequences, which improves the accuracy of the approximate distance, during the course of query processing.

Figure 7 shows the gradual 'refinement' of the approximation. In Figure 7 (a), we compute the approximate distance from the coarsest version of sequences as the first step of the refinement. If the distance is greater than d_{cb}, we can safely prune P. Otherwise, we compute the distance from the more accurate sequences as the second refinement step (See Figure 7 (b)). We need to compute the exact distance from the original sequences only if the approximate distance does not exceed d_{cb} (See Figure 7 (c)).

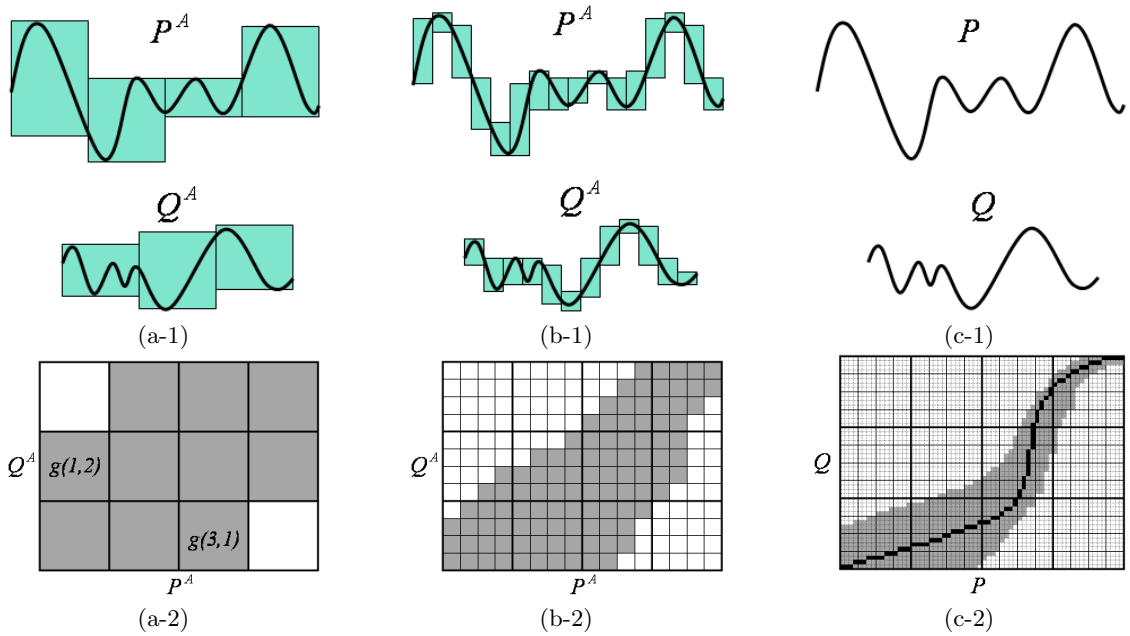


Figure 7: Sketch of FTW. The bottom row shows the time warping matrices; the gray squares are potentially useful while the white squares are not promising. The black squares show the final optimal path. (a) The coarse approximation requires a low computation cost. (b) The refinement gives higher approximation quality. (c) The exact distance calculation by EarlyStopping that visits the small gray area.

3.3 Details for the proposed LBS

We first introduce the approximate segments of sequences. Let $P = \{p_1, p_2, \ldots, p_N\}$ be a sequence of length N, then the i-th approximate segment p_i^A of P is defined as follows:

$$p_i^A = \{p_i^R, p_i^T\}, \qquad p_i^R = \{p_i^L, p_i^U\}$$
$$p_i^L = \min\{p_x, \ldots, p_y\}, \quad p_i^U = \max\{p_x, \ldots, p_y\}$$
$$x = \begin{cases} 1 & (i = 1) \\ \sum_{j=1}^{i-1} p_j^T + 1 & (2 \le i \le n), \end{cases} \qquad y = \sum_{j=1}^{i} p_j^T$$

where p_i^R and p_i^T denotes the segment range and time interval of P, p_i^L and p_i^U are the lower and upper bounds of p_i^R. That is, p_i^L and p_i^U are the maximum and minimum values among the p_i^T elements within the subsequence from p_x to p_y. Therefore, we approximate P by:

$$P^A = \{p_1^A, p_2^A, \ldots, p_n^A\}$$

where n denotes the number of segments.

EXAMPLE 1. Let P be a data sequence of length $N = 18$, and $p_i^T = 3$ $(i = 1, \ldots, 6)$. The approximate segments of P can be derived as follows:

$$P = \{3, 2, 3, 5, 7, 6, 6, 5, 7, 10, 12, 11, 13, 15, 14, 12, 13, 12\}$$
$$p_1^R = \{2, 3\},\ p_2^R = \{5, 7\},\ p_3^R = \{5, 7\},$$
$$p_4^R = \{10, 12\},\ p_5^R = \{13, 15\},\ p_6^R = \{12, 13\}$$

Let Q be a query sequence of length $M = 12$, and $q_i^T = 3$ $(i = 1, \ldots, 4)$ The approximate segments of Q is computed as:

$$Q = \{7, 6, 7, 10, 12, 11, 9, 8, 11, 10, 8, 9\}$$
$$q_1^R = \{6, 7\},\ q_2^R = \{10, 12\},\ q_3^R = \{8, 11\},\ q_4^R = \{8, 10\}$$

We propose a new lower bounding distance measure based on a combination of dynamic programming and approximate segments. Let $P^A = \{p_1^A, \ldots, p_n^A\}$ and $Q^A = \{q_1^A, \ldots, q_m^A\}$. We then introduce LBS (Lower Bounding distance measure with Segmentation) as follows:

$$D_{lbs}(P^A, Q^A) = g(n, m)$$
$$g(i, j) = g_{cell}(i, j) + \min \begin{cases} g(i, j-1) \\ g(i-1, j) \\ g(i-1, j-1) \end{cases} \qquad (2)$$
$$g_{cell}(i, j) = \min(p_i^T, q_j^T) \cdot D_{seg}(p_i^R, q_j^R)$$
$$g(0, 0) = 0, \quad g(i, 0) = g(0, j) = \infty$$

where $D_{seg}(p_i^R, q_j^R)$ denotes the distance between p_i^R and q_j^R; the distance of two ranges is intuitively the distance of their two closest points. Formally:

$$D_{seg}(p_i^R, q_j^R) = \begin{cases} \|p_i^L - q_j^U\| & (p_i^L > q_j^U) \\ \|q_j^L - p_i^U\| & (q_j^L > p_i^U) \\ 0 & (otherwise) \end{cases}$$

Notice that, if each range degenerates to a point, the distance D_{seg} becomes the original distance $\|*, *\|$ between two points.

THEOREM 1. *Let P^A and Q^A be the approximate segments of sequences P and Q, respectively, then*

$$D_{lbs}(P^A, Q^A) \leq D_{dtw}(P, Q) \tag{3}$$

PROOF. Since

$$g_{cell}(1,1) = \min(p_1^T, q_1^T) \cdot D_{seg}(p_1^R, q_1^R)$$

the following inequalities hold:

$$g(1,1) \leq f(x, q_1^T) \qquad (1 \leq x \leq p_1^T)$$
$$g(1,1) \leq f(p_1^T, y) \qquad (1 \leq y \leq q_1^T)$$

Since

$$g_{cell}(i,j) = \min(p_i^T, q_j^T) \cdot D_{seg}(p_i^R, q_j^R)$$

we have

$$\min\{g(i-1,j-1),\ g(i-1,j)\} \leq f(x,y) \quad (x = \sum_{k=1}^{i-1} p_k^T,\ \sum_{k=1}^{j-1} q_k^T < y \leq \sum_{k=1}^{j} q_k^T)$$
$$\min\{g(i-1,j-1),\ g(i,j-1)\} \leq f(x,y) \quad (\sum_{k=1}^{i-1} p_k^T < x \leq \sum_{k=1}^{i} p_k^T,\ \ y = \sum_{k=1}^{j-1} q_k^T)$$

Therefore, we obtain

$$g(i,j) \leq f(x,y) \quad (x = \sum_{k=1}^{i} p_k^T,\ \ y = \sum_{k=1}^{j} q_k^T)$$

Thus, $g(n,m) \leq f(N,M)$, which completes the proof. □

3.4 EarlyStopping

To compute $D_{lbs}(P^A, Q^A)$ efficiently, we introduce an algorithm called EarlyStopping (See Figure 8). During search processing, we maintain the best-so-far distance, d_{cb}; lower bounding distances greater than d_{cb} do not need to be computed using Equation (2). EarlyStopping reduces the distance computation cost by using d_{cb}.

EXAMPLE 2. Let us consider the approximate segments of P and Q as shown in Example 1. The lower bounding distance between P and Q is efficiently computed by using EarlyStopping as shown in Figure 9. Each value indicates $g(i,j)$ in the figure. If $d_{cb} = 28$, the calculations of $g(1,3)$ and $g(1,4)$ are omitted since $g(1,2) = 116$ is greater than d_{cb}. Similarly, the other white cells are also omitted since $g(1,2) > d_{cb}$ and $g(2,2) = g(3,2) = g(4,1) = 36 > d_{cb}$. As a result, the lower bounding distance between P and Q is computed as $g(6,4) = 30$. Consequently, we can safely prune P, since the lower bounding distance is greater than d_{cb}.

EarlyStopping can be applied to the exact time warping distance calculation for data sequences as well as for the approximate distance calculation, and it reduces computation cost. Thus, we utilize EarlyStopping for computing the exact distance $D_{dtw}(P,Q)$ of the sequences P and Q. Moreover, EarlyStopping supports global constraints [22]. When the warping scope is limited by global constraints, we change the initial values of *begin*[*i*] and *end*[*i*] in Figure 8 according to the given warping scope.

```
Algorithm EarlyStopping(P^A, Q^A, d_cb)
  // Set the initial values of begin[i] and end[i] according
  //                                  to the global constraint
  for i = 1 to n do
    begin[i] := 1;
    end[i] := m;
  endfor
  // Compute the lower bounding distance
  for i = 1 to n do
    for j = begin[i] to end[i] do
      compute g(i, j);
      if i ≠ 1 and i ≠ n then
        if j > end[i − 1] and g(i, j) > d_cb then
          end[i] := j;
          break;
        endif
      endif
    endfor
    if no grid cell satisfies g(i, j) ≤ d_cb then
      return g(i, end[i]);
    else
      begin[i] := min{j|g(i, j) ≤ d_cb};
      end[i] := max{j|g(i, j) ≤ d_cb};
      if i ≠ n and begin[i + 1] < begin[i] then
        begin[i + 1] := begin[i];
      endif
    endif
  endfor
  return g(n, m);
```

Figure 8: Time warping distance calculation algorithm using the k-nearest neighbor distance.

3.5 Refinement

In the preceding section, we presented an algorithm computing an approximate distance with approximate segments of a single granularity. However, we can use segments of multiple granularities for query processing. Here, we describe the gradual refinement of the distance approximation with multiple granularities. We use c different versions of P for query processing. Let P_i^A be the coarse version of P, which is computed from the time interval t_i as:

$$1 < t_1 < t_2 < \ldots < t_{c-1} < t_c < N$$

That is, P_c^A is the coarsest, while P_1^A is the most accurate.

Various algorithms have been proposed to find the optimal representation of approximate segments of each sequences (e.g., [7]). We will use equal-sized segments to approximate sequences for simplicity although segments of different time intervals would be acceptable to FTW. Since we operate on equal-sized segments, the time interval of each i-th segment is

$$p_i^T = \begin{cases} t & (1 \leq i \leq n-1) \\ tn - N & (i = n) \end{cases} \tag{4}$$

where $n = \lceil N/t \rceil$.

3.5.1 Indexing

We propose a sequential structure to accelerate search performance. The coarse version of P, P^A, consists of the series of time intervals P^T and the series of segment ranges P^R. While the search algorithm uses both P^T and P^R, the data structure does not includes P^T since P^T can be easily com-

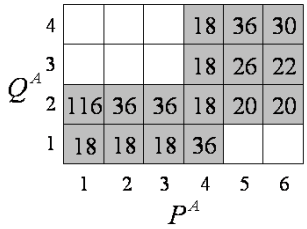


Figure 9: Example of a time warping distance calculation using approximate segments. The white cells are omitted because they are greater than $d_{cb} = 28$.

Algorithm Search(Q, k)
  compute $Set(Q^A)$; // $Set(Q^A) = \{Q_c^A, \ldots, Q_1^A\}$
  // Collect k sequences as the initial candidates
  for each $P \in database$ **do**
    $d_{coarse}[P] := D_{lbs}(P_r^A, Q_r^A)$;
    if $d_{coarse}[P] < TempList[k].dist$ **then**
      add P and $d_{coarse}[P]$ to $TempList$, and sort;
  endfor
  // Compute the initial k-th nearest neighbor distance
  for each $P \in TempList$ **do**
    // NNL is the sorted nearest neighbor list
    add P and $D_{dtw}(P, Q)$ to NNL, and update d_{cb};
  // Search for the k-nearest neighbors
  for each $P \in database$ **do**
    if $P \notin TempList$ **and** $d_{coarse}[P] \leq d_{cb}$ **then**
      for $i := c - 1$ **to** 1 **do**
        d_{approx} := EarlyStopping(P_i^A, Q_i^A, d_{cb});
        if $d_{approx} > d_{cb}$ **then**
          break;
      endfor
      if $d_{approx} \leq d_{cb}$ **then**
        d_{exact} := EarlyStopping(P, Q, d_{cb});
        if $d_{exact} \leq d_{cb}$ **then**
          add P and d_{exact} to NNL, and update d_{cb};
        endif
      endif
    endif
  endfor
  return NNL;

Figure 10: k-nearest neighbor search algorithm.

puted by using Equation (4). The data structure is simply an array of feature data of a sequence P:

$$F(P) = \{N, Set(P^R)\}$$
$$Set(P^R) = \{P_c^R, \ldots P_1^R\}$$

The feature data $F(P)$ consists of the length of P, N, and the set of segment ranges of P, $Set(P^R)$.

3.5.2 Search Algorithm

Our search algorithm is based on the following ideas:

1. *The search algorithm first collects as candidates the k-nearest neighbors based on the lower bounding distance measure computed from approximate segments.*

   FTW reduces computation cost by using the current k-th nearest neighbor distance, d_{cb}; consequently its efficiency is improved if the current value of d_{cb} is close to the final k-th nearest neighbor distance early in the search process. Therefore, the algorithm calculates the lower bounding distance between the query sequence and each data sequence from their approximate segments, and then collects the k sequences that have the shortest distance in the sense of the lower bounding distance. The set of the k collected sequences is regarded as the initial candidate set for the k-nearest neighbor query. We obtain the initial k-th nearest neighbor distance by computing the exact time warping distance between the collected sequences and the query sequence. We pick the lowest granularity (i.e., the coarsest sequences) for this initial processing.

2. *The search algorithm gradually enhances the accuracy of the distance approximations.*

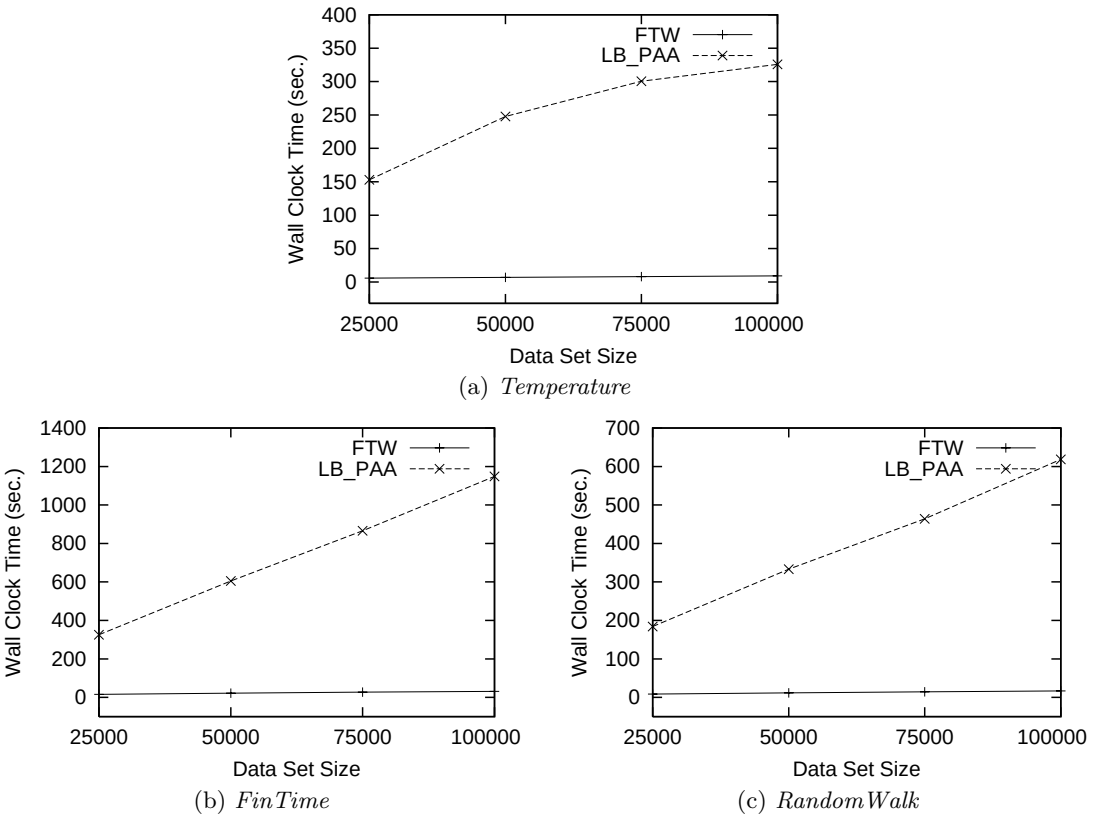
   Distance approximations incur a lower computation cost than exact distance calculations; moreover, we require a lower distance computation cost for coarser sequences. The search algorithm first calculates an approximate distance of the coarsest data sequence of the time interval t_c. If the approximate distance is greater than d_{cb}, the algorithm excludes the data sequence without computing its exact distance. If the approximate distance does not exceed d_{cb}, the algorithm then generates a more accurate approximation based on t_{c-1}, and compares it with d_{cb}. That is, the search algorithm gradually enhances the accuracy of the distance approximations. It determines the exact distance only when the approximate distance derived from t_1 does not exceed d_{cb}.

Figure 10 shows the search algorithm that uses the data structure described in Section 3.5.1. We first compute the approximate segments of a given query sequence Q. We then collect the candidate k-nearest neighbor sequences based on the lower bounding distance measure. The set of the k collected sequences is used as the initial candidate set. We obtain the initial k-th nearest neighbor distance by calculating the exact time warping distance between each candidate sequence and the query sequence. We compute a tighter approximation distance, while reducing the time interval. If the lower bounding distance is greater than d_{cb}, we exclude the data sequence since it can't be one of the k-nearest neighbors.

Although we described only the search algorithm for k-nearest neighbor queries, FTW can be applied to range queries. It utilizes the current k-th nearest neighbor distance d_{cb} for k-nearest neighbor queries, and the search range is used to handle range queries.

4. EXPERIMENTS

We conducted experiments to verify the superiority of FTW. [14] claims that the method of [14] outperforms those of [26] and [17]; moreover, [27] confirms that Zhu et al.'s method is more effective than that of [14]. Thus, we show the results of a performance evaluation that compared FTW with the best existing method proposed in [27]. [27]'s method is denoted by LB_PAA. For LB_PAA, we used the R*-tree [3], and each sequence was indexed according to 16 reduced dimensions as shown in [27]. We evaluated the search performance mainly



(a) *Temperature*

(b) *FinTime*

(c) *RandomWalk*

Figure 11: Wall clock time as a function of data set size ($N = 2048$). The superiority of FTW grows as data size increases.

based on wall clock time since the wall clock time for DTW exceeds the disk access time for retrieval processing; that is, the total execution time depends primarily on wall clock time. The wall clock time was measured on an Intel Xeon 2.8GHz with 1GB of memory, running Linux. We used 20-nearest neighbor queries. Each result reported here for a particular database size and sequence length is the average of 100 trials.

The data sets we used included the following real and synthetic data sets:

1. *Temperature*: Temperature measurements, from 55 sensors in buildings of Carnegie Mellon University. Each sensor gives one value every 30 seconds.
2. *FinTime*: This is the financial time-series benchmark from [11]. We used historical stock market data for 100,000 securities.
3. *RandomWalk*: We generated 100,000 sequences by using random-walk models [25]: $p_i = p_{i-1} + x_i$ where the p_1 of each sequence is uniformly distributed in the range (0, 10). x_i is normally distributed and the variance is 1.

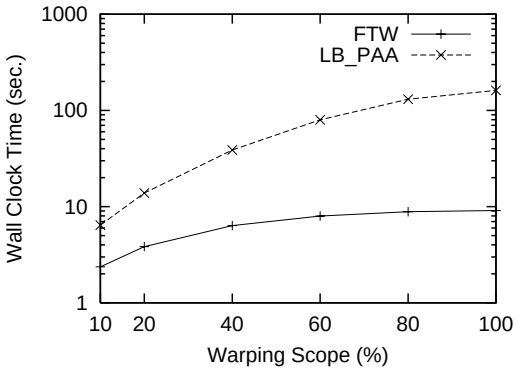
For FTW, we used four different time intervals ($t_1 = 2$, $t_2 = 8$, $t_3 = 32$, $t_4 = 128$) for sequences of length 2048. For sequences of length 512, we used three varieties (t_1, t_2 and t_3). Before discussing the search performance, we should mention the cost of constructing the index structures. Our method required 381 seconds to construct the data structure for the data set of sequence length 2048 and size 100,000. LB_PAA required 531 seconds including the time for constructing the R*-tree.

4.1 Search Performance

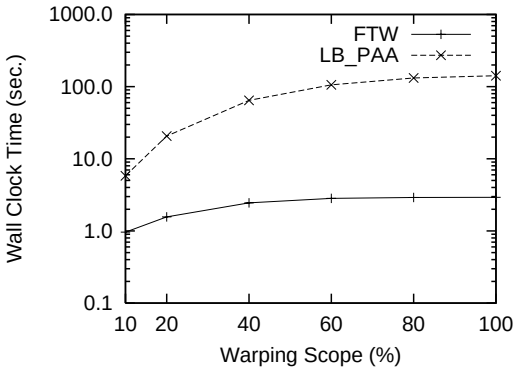
To evaluate the search performance, we compared FTW with LB_PAA. We constructed index structures for the data sets of *Temperature*, *FinTime*, and *RandomWalk*.

We present experimental results on search performance for when the data set size varies. Figure 11 depicts the wall clock time of FTW and LB_PAA for various data set sizes. The global constraint we employed for this figure is the Itakura Parallelogram, which is widely used in practical situations [5, 22]. The exact distance calculation also employed the same global constraint. Database size varied from 25,000 to 100,000. Note that the y-axis starts from a negative value to avoid the overlap between the x-axis and the line of FTW. The experimental results show that FTW gives superior approximations for all data sets. As the database size increases, the effectiveness of FTW increases, making it even more attractive for large data sets.

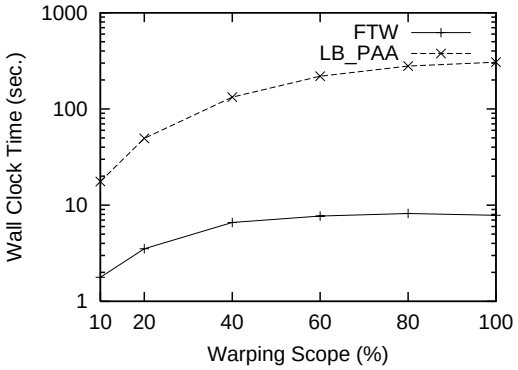
Figures 12 and 13 compare the search methods in terms of



(a) *Temperature*

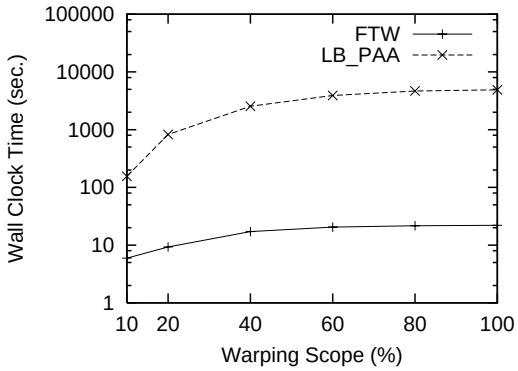


(b) *FinTime*

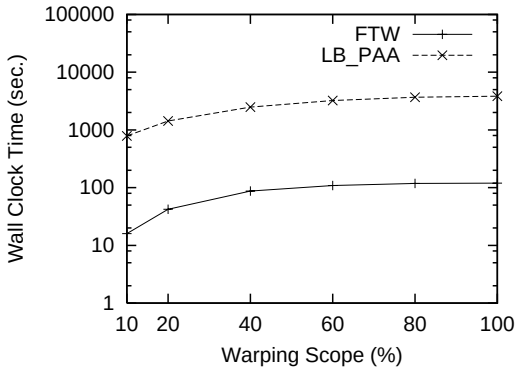


(c) *RandomWalk*

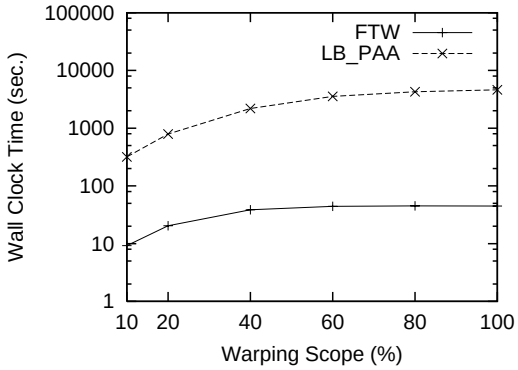
Figure 12: Wall clock time as a function of warping scope ($N = 512$). Note that FTW is up to 49 times faster than the best existing method.



(a) *Temperature*



(b) *FinTime*



(c) *RandomWalk*

Figure 13: Wall clock time as a function of warping scope ($N = 2048$). Note that FTW is up to 222 times faster than the best existing method.

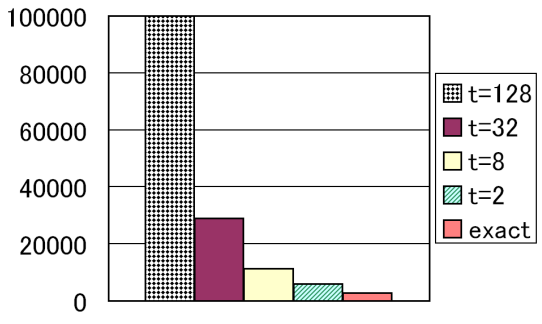


Figure 14: Frequency of approximation use (*FinTime*, $N = 2048$).

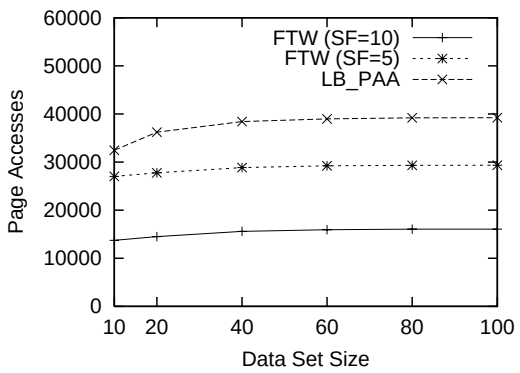


Figure 15: Number of page accesses as a function of data set size (*FinTime*, $N = 2048$).

wall clock time. We employed the Sakoe-Chiba Band for these figures, and let the width of the warping scope vary from 10% to 100% of the sequence length N. In these figures, the data set size is 100,000, and the scale of the y-axis is logarithmic. These figures indicate that FTW significantly reduces the search cost for all data sets. FTW outperforms LB_PAA significantly, by orders of magnitude. Concretely, FTW is up to 222 times faster than LB_PAA.

Figure 14 shows how often each approximation was used in FTW for the data set size of 100,000. As shown in the figure, most of the data sequences are excluded by the approximations of $t_4 = 128$ and $t_3 = 32$. The approximations of t_{i-1} incur a computation cost that is $(t_i/t_{i-1})^2$ times that of the cost of t_i; moreover, the exact time warping distance calculation requires a higher computation cost than the approximation of t_1. The coarser approximation provides reasonable approximation quality, and its calculation speed is high. On the other hand, although the approximation with higher granularity is not very fast, it offers good approximation quality. Accordingly, using approximations of various granularities has significant advantages in terms of approximation quality and calculation speed. FTW efficiently prunes a large number of the search candidates, which leads to a significant reduction in the search cost.

Figure 15 shows the number of page accesses for FTW and LB_PAA. Our data structure is simply an array of feature data. Although data sequences are accessed randomly, feature data are visited sequentially in query processing. A sequential scan of feature data should significantly boost performance because of the sequential nature of their I/O requests. In [23], Weber et al. indicate two speed-up factors for the phenomenon: a practical factor of 10 and a conservative one of 5. Thus, we used these speed-up factors, SF, in our experimental evaluations. The number of page accesses of FTW for a speed-up factor SF is given by:

$$PA_{SF} = \frac{PA_{fd}}{SF} + PA_{ds},$$

where PA_{fd} and PA_{ds} are the number of page accesses for feature data and data sequences, respectively. Figure 15 indicates that FTW outperforms LB_PAA for $SF = 5$ and $SF = 10$ in terms of page accesses.

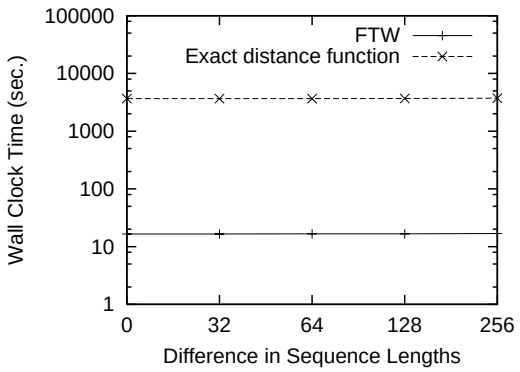


Figure 16: Wall clock time as a function of the difference in sequence lengths in a data set (*RandomWalk*, $N = 2048$).

4.2 Search Performance for Sequences of Different Lengths

We created four sequence data sets that contained sequences with different lengths, N: *Random*(2048, 32), *Random*(2048, 64), *Random*(2048, 128), and *Random*(2048, 256), where *Random*(N_{ave}, N_{diff}) is a random function, N_{ave} is the average sequence length in a data set, and N_{diff} is the difference between the minimum sequence length and the maximum sequence length in a data set. All data sets had 100,000 sequences.

Figure 16 shows the wall clock time of a linear scan and FTW as a function of N_{diff}. We employed the Itakura Parallelogram for this figure. Clearly, FTW outperforms sequential scanning for all data sets. Its search performance is superior even when N_{diff} is large.

5. CONCLUSIONS

We presented a new search method for DTW, called FTW, which meets all the requirements mentioned in the introduction. FTW has been careful designed based on a new lower bounding distance measure that approximates the time warping distance. Based on this lower-bounding idea, and sev-

eral careful optimizations ('EarlyStopping', 'Refinement'), our method has *all* of the following specifications, that no competing method matches:

1. It is fast: In experiments on real and synthetic data sets, FTW clearly outperformed the best existing method, for all queries, achieving *one or two orders of magnitude* speed up.
2. It has no false dismissals (by our Theorem 1)
3. It can handle sequences of arbitrary lengths
4. It allows for arbitrary warping restrictions, as well as *no* restrictions at all

Our experimental results reveal that FTW is significantly faster than the best existing method, outperforming it by at least *one order of magnitude*, and occasionally up to 222 times.

A promising but very challenging research direction is to extend FTW for a streaming setting. The goal would be to monitor time sequences, looking for one or more prespecified patterns, where the (dis-)similarity score is determined by DTW.

6. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Proceedings of the 4th Conference on Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, February 1993.

[2] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceedings of VLDB*, pages 490–501, September 1995.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of ACM SIGMOD*, pages 322–331, May 1990.

[4] S. Berchtold, C. Böhm, and H.-P. Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. In *Proceedings of ACM SIGMOD*, pages 142–153, June 1998.

[5] D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248, AAAI/MIT, 1996.

[6] S. Chu, E. Keogh, D. Hart, and M. Pazzani. Iterative deepening dynamic time warping for time series. In *Proceedings of SIAM International Conference on Data Mining*, 2002.

[7] C. Faloutsos, H. V. Jagadish, A. O. Mendelzon, and T. Milo. A signature technique for similarity-based queries. In *SEQUENCES*, June 1997.

[8] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of ACM SIGMOD*, pages 419–429, May 1994.

[9] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[10] R. W. Hamming. *Digital Filters*. Englewood Cliffs, N. J., 1977.

[11] K. J. Jacob and D. Shasha. Fintime — a financial time series benchmark. http://cs.nyu.edu/cs/faculty/shasha/fintime.html, March 2000.

[12] J.-S. R. Jang and H.-R. Lee. Hierarchical filtering method for content-based music retrieval via acoustic input. In *Proceedings of ACM Multimedia*, pages 401–410, September/October 2001.

[13] H. Kawasaki, T. Yatabe, K. Ikeuchi, and M. Sakauchi. Automatic modeling of a 3d city map from real-world video. In *Proceedings of ACM Multimedia (1)*, pages 11–18, October/November 1999.

[14] E. J. Keogh. Exact indexing of dynamic time warping. In *Proceedings of VLDB*, pages 406–417, August 2002.

[15] E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proceedings of ACM SIGMOD*, pages 151–162, May 2001.

[16] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, pages 263–286, 2000.

[17] S.-W. Kim, S. Park, and W. W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of ICDE*, pages 607–614, April 2001.

[18] Y.-S. Moon, K.-Y. Whang, and W.-S. Han. General match: a subsequence matching method in time-series databases based on generalized windows. In *Proceedings of ACM SIGMOD*, pages 382–393, June 2002.

[19] D. W. Mount. *Bioinfomatics: Sequence and Genome Analysis*. Cold Spring Harbor, New York, 2000.

[20] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Englewood Cliffs, N. J., 1975.

[21] K. Otsuka, T. Horikoshi, S. Suzuki, and H. Kojima. Memory-based forecasting for weather image patterns. In *Proceedings of the 17th Conference on Artificial Intelligence (AAAI)*, pages 330–336, July 2000.

[22] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs, N. J., 1993.

[23] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of VLDB*, pages 194–205, August 1998.

[24] M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. A K Peters Ltd, Massachusetts, 1994.

[25] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *Proceedings of VLDB*, pages 385–394, September 2000.

[26] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of ICDE*, pages 201–208, February 1998.

[27] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *Proceedings of ACM SIGMOD*, pages 181–192, June 2003.