

Efficient Retrieval of Similar Time Sequences under Time Warping

Byoung-Kee Yi

H.V. Jagadish

Christos Faloutsos*

Dept. of Computer Science
University of Maryland
College Park, MD 20742
kee@cs.umd.edu

AT&T Labs
Florham Park, NJ 07932
jag@research.att.com

Dept. of Computer Science
University of Maryland
College Park, MD 20742
christos@cs.umd.edu

Abstract

Fast similarity searching in large time-sequence databases has typically used Euclidean distance as a dissimilarity metric. However, for several applications, including matching of voice, audio and medical signals (e.g., electrocardiograms), one is required to permit local accelerations and decelerations in the rate of sequences, leading to a popular, field-tested dissimilarity metric called the “time warping” distance.

From the indexing viewpoint this metric presents two major challenges: (a) it does not lead to any natural indexable “features”, and (b) comparing two sequences requires time quadratic in the sequence length. To address each problem, we propose to use: (a) a modification of the so-called “FastMap”, to map sequences into points, with little compromise of “recall” (typically zero), and (b) a fast, linear test, to help us discard quickly many of the false alarms that FastMap will typically introduce. Using both ideas in cascade, our proposed method achieved up to an order of magnitude speed-up over sequential scanning on both real and synthetic datasets.

1 Introduction

A doctor watching an electrocardiogram is often looking for a pattern that is indicative of a problem. We would like for a computer to *watch* the readings on an electrocardiograph, and to cause an alert, or take other appropriate action, when a pattern is observed that is characteristic of a particular type of heart failure. In database terms, a query response is expected when a given sequence *approximately* matches any one of several patterns in a database.

Applications of approximate sequence matching abound: in financial time sequences (“find stocks that move like Microsoft”); digital audio/voice clips (“find clips that sound like a given person”) [1, 4]; scientific databases (“find times in the past that had similar solar magnetic wind patterns with the ones today” [13]).

In the area of speech recognition, this problem has been studied extensively, and is called the “(dy-

namic) time warping”. Virtually all speech recognition systems speed-up and slow down portions of the speech samples to be matched. Standard techniques to accomplish this use dynamic programming, with quadratic complexity (*i.e.*, proportional to the product of the lengths of the sequences being matched).

The same ideas could be used for matching in a database context, but are likely to prove too expensive. We would like to have a very fast matching technique, and ideally even an indexing technique for this purpose.

In this paper we propose two such techniques. The first technique is based on FastMap [3]. The idea is to make use of the given distance measures to map sequences into points in k -d space, and then build an index structure. The other technique is to determine a cheaply computed lower bound on the original distance function, that can be used as a filter to discard non-qualifying sequences quickly.

The rest of the paper is organized as follows. Section 2 provides a survey on related works. In Section 3, we lay out the basic framework and define the problem under study. In Section 4, we present the two proposed techniques in detail. We also discuss how to combine the two techniques, as well as some variants of the basic techniques. In Section 5, we present empirical results comparing the performance of the techniques. Finally, Section 6 concludes this work.

2 Related Work

Similarity-based matching of time sequences has been studied extensively in the signal processing area, and specifically in speech processing. However, the usual assumptions are a small dataset (*e.g.*, a few tens of phonemes) so that the primary concern is precision rather than efficiency in the presence of large datasets.

Speed is the main focus in the recent database work on sequence matching. In [1], we examined the Euclidean distance, and suggest using the Discrete Fourier Transform (DFT). We argued that most of real signals need only a few DFT coefficients to approximate them. Then, we proposed an indexing mechanism called *F-Index* which takes a few first coefficients and regards them as a point in the Euclidean space, hence it makes possible to use readily available *Spatial Access Methods* (SAMs). The proposed method

*Also with Inst. for Systems Research, University of Maryland. Currently on leave at Carnegie Mellon University, Pittsburgh, PA. This research was partially supported by the NSF under Grants No. EEC-94-02384, IRI-9205273 and IRI-9625428.

may allow a few false alarms which can be removed in the post-processing stage, but guarantees no false dismissals. This method was proposed for matching sequences of equal length. In [4] we generalized the approach for subsequence matching.

Follow-up work by Goldin and Kanellakis [5] suggested that we normalize the sequences first, to allow for differences in level and scale.

All the above approaches assume Euclidean distance as the underlying similarity measure. Agrawal *et al* [2] introduce a new distance function for time sequences, aiming to capture the intuitive notion that two sequences should be considered similar if they have enough non-overlapping time-ordered pairs of similar subsequences. The model allows the amplitude of one of the two sequences to be scaled by any suitable amount and its offset adjusted appropriately. It also allows non-matching gaps in the matching subsequences.

Rafei and Mendelzon [9] extend previous work by proposing techniques to handle moving average and time scaling (*i.e.*, globally stretching or shrinking of the time axis), but not time warping.

In [6], we developed a domain independent framework for defining queries in terms of similarity of objects. Our framework has three components: a pattern language, a transformation rule language, and a query language. The framework can be *tuned* to the needs of a specific application domain, such as time sequences, molecules, text strings or images, by the choice of these languages.

Sheshadri *et al* [12] suggest a new data model and an algebraic language for sequences in general. They also propose a sophisticated optimization technique, but do not mention about similarity among sequences and query processing technique based on similarity.

A topic that none of the above articles has tackled is the problem of indexing, when local, time-warping transformations are allowed. This is a difficult problem, because the DFT methods of [1, 4] do not work any more. This is exactly the focus of the rest of this work.

3 Background

We assume that all our sequences are real-valued samples obtained at equi-spaced points in time. We denote a sequence $\langle x_1, \dots, x_n \rangle$ as \vec{x} . Table 1 gives a list of symbols used in the rest of the paper.

For two sequences $\vec{x} = \langle x_1, \dots, x_n \rangle$, $\vec{y} = \langle y_1, \dots, y_n \rangle$ the \mathcal{L}_p distance is defined as follows:

$$\mathcal{D}_p(\vec{x}, \vec{y}) = \sum_{i=1}^n |x_i - y_i|^p$$

For $p = 1$ this reduces to the “Manhattan” or “city-block” distance; for $p = 2$ it becomes the popular Euclidean distance.

3.1 The Time-Warping Transformation

Given a sequence $\vec{x} = \langle x_1, \dots, x_n \rangle$, let $Head(\vec{x})$ denote x_1 and $Rest(\vec{x})$ denote $\langle x_2, \dots, x_n \rangle$. Also let $stutter_i(\vec{x}) = \langle x_1, \dots, x_{i-1}, x_i, x_i, x_{i+1}, \dots, x_n \rangle$, the

Symbol	Definition
\mathcal{D}_p	\mathcal{L}_p -based distance function
\mathcal{D}_{base}	base distance function, <i>e.g.</i> , \mathcal{D}_1 or \mathcal{D}_2
\mathcal{D}_{warp}	time warping distance
\mathcal{D}_{lb}	distance function to lower-bound \mathcal{D}_{warp}
\vec{x}	time sequence
$\langle \rangle$	null time sequence
x_i	i -th element of \vec{x}
$ \vec{x} $	length of \vec{x}
$Head(\vec{x})$	the first element of \vec{x}
$Rest(\vec{x})$	the rest of \vec{x} but the first
N	database size
k	dimension in a Euclidean space
ϵ	tolerance in range query

Table 1: List of symbols.

$n + 1$ long sequence that repeats x_i and shifts all later elements to the right.

Definition 1 *The time warping distance between two sequences is defined, following [8], as:*

$$\begin{aligned} \mathcal{D}_{warp}(\langle \rangle, \langle \rangle) &= 0, \\ \mathcal{D}_{warp}(\vec{x}, \langle \rangle) &= \mathcal{D}_{warp}(\langle \rangle, \vec{y}) = \infty, \\ \mathcal{D}_{warp}(\vec{x}, \vec{y}) &= \mathcal{D}_{base}(Head(\vec{x}), Head(\vec{y})) + \\ &\min \left\{ \begin{array}{ll} \mathcal{D}_{warp}(\vec{x}, Rest(\vec{y})), & (x\text{-stutter}) \\ \mathcal{D}_{warp}(Rest(\vec{x}), \vec{y}), & (y\text{-stutter}) \\ \mathcal{D}_{warp}(Rest(\vec{x}), Rest(\vec{y})) & (no\ stutter) \end{array} \right\} \end{aligned}$$

where $\langle \rangle$ denotes a null sequence. \mathcal{D}_{base} can be any of the distance functions defined previously, although our primary concern is with \mathcal{D}_1 , or the city-block distance. Also note that this definition does not require two sequences to be of the same length. In the case of time warping distance, we allow as many stuttering as needed at no cost.

Defined as a recurrence, the time-warping distance can be computed by a dynamic programming algorithm whose complexity is $O(|\vec{x}| \times |\vec{y}|)$. See [11] and [8] for more details and other variants of the basic algorithm.

Figure 1 shows two time sequences, before and after the time warping. The sequences are mixtures of similar harmonics: $x(t) = 10 \sin(0.5t) + 5 \sin(0.25t)$ and $y(t) = 11 \sin(0.55t) + 4.5 \sin(0.26t)$ respectively. Observe how time warping automatically adjusts peaks and valleys of two sequences.

Proposition 1 $\mathcal{D}_{warp}()$ does not satisfy the triangle inequality.

Proof: By counter-example, consider $\vec{x} = \langle 0 \rangle$, $\vec{y} = \langle 1, 2 \rangle$, and $\vec{z} = \langle 1, 2, 2 \rangle$. Then, we have

$$\mathcal{D}_{warp}(\vec{x}, \vec{z}) > \mathcal{D}_{warp}(\vec{x}, \vec{y}) + \mathcal{D}_{warp}(\vec{y}, \vec{z}).$$

which completes the proof. \square

This fact has significant implication on the method we can use for indexing: *Any* indexing technique

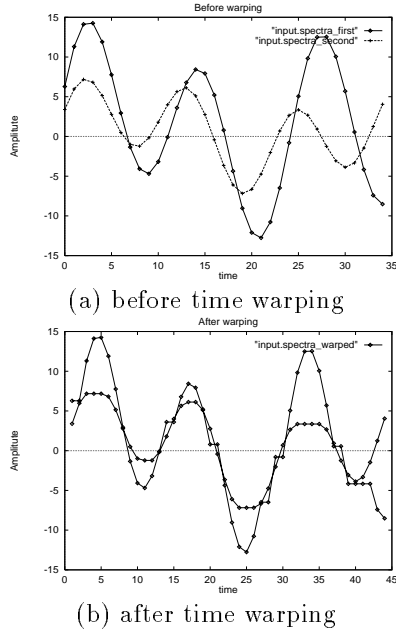


Figure 1: Illustration of two similar sequences, before and after time warping.

that assumes the triangle inequality implicitly or explicitly, cannot avoid producing false dismissals. In other words, all spatial access methods, as well as all the methods that use distance/metric/vantage-point trees, cannot avoid false dismissals. The only method that guarantees no false dismissals is sequential scanning, which is prohibitive for a large collection of long sequences. So our goal is to resolve, rather than eliminate, false dismissals.

4 Proposed Techniques

Consider a database containing many time sequences of arbitrary length. We wish to find all sequences similar to a given query sequence, that is, within ϵ time-warping distance.

A straightforward way to process such query is to scan all sequences and compute $\mathcal{D}_{warp}()$ for each scanned sequence, to select those that qualify. While very simple, it can be very slow, because,

- it reads every sequence in the database (and thus scales poorly), and
- it computes the (expensive) time warping distance from each sequence of the database.

What is unique in this problem is that not only I/O cost (the first case) matters, but also computation cost (the second case) does. Consequently, any promising techniques should address both of these problems.

To solve these problems, we propose the following techniques.

- To use FastMap to build index structure to speed up query processing. This technique may result

```

algorithm fastmap-range-search
   $R := \{\}$ ; /* response set */
  /* filtering step */
  Given  $\vec{q}$ , foreach sequence  $\vec{s}_i$ 
  in the database  $S$ ,
    if  $(\mathcal{D}_2(F(\vec{q}), F(\vec{s}_i)) \leq \epsilon)$ ,
      then add  $i$  to  $R$ ;
  /* post-processing step */
  Foreach  $i$  in  $R$ ,
    if  $(\mathcal{D}_{warp}(\vec{q}, \vec{s}_i) > \epsilon)$ ,
      then remove  $i$  from  $R$ ;
  Report  $R$ ;
end algorithm

```

Algorithm 1: ϵ -Range Query based on FastMap.

in some few false dismissals. We will also discuss how to reduce false dismissals.

- To use a new distance function that uniformly underestimate time warping distance. This approach guarantees *no* false dismissals.
- To use the combination of the two techniques. Since the two techniques are independent of each other, they can be combined in a pipelined manner. The parameters of the first step, FastMap, can be set to minimize the chance of false dismissals, even at the cost of more false positives, relying upon the second step that follows to weed most of these out.

In the subsequent sections, we describe precisely the proposed techniques.

4.1 FastMap-Based Technique

The first technique we propose is based on a method called “FastMap” [3]. It works as follows: Given N objects and a distance function, it maps the objects into N points in a k - d space, so that the original distances are preserved well. The parameter k may be given by the user or can be tuned for better system performance in our application. The key idea is to pretend as if objects are indeed points in some unknown, n -dimensional space, and try to project these points on k mutually orthogonal directions, using only the distance information.

After the objects are mapped into k - d points, we can use any spatial access method to organize them and to search for range queries. FastMap is linear on the number N of objects (*i.e.*, sequences). Moreover, it takes $O(k)$ time to map a query sequence into a k - d point, that is, the time is constant with respect to the database size N .

Like any other methods (see Proposition 1), FastMap may introduce false dismissals, if the triangle inequality is not obeyed. We observed that we can avoid more false dismissals, if we use square root of the original distances. Thus, we use this technique for the rest of this work.

Algorithm 1 describes how range queries are handled using FastMap. If FastMap is applied on the

square rooted distances, the search range should also be square rooted. Note that $F(\vec{s})$ denotes the $k-d$ coordinates of a sequence \vec{s} . In the filtering step, two sequences are compared in terms of $k-d$ Euclidean distance rather than the time warping distance. Irrelevant sequences are filtered out at this step. Some non-qualifying sequences may be included, but those are removed in the post-processing step.

Algorithm 1 is faster than the naive method for several reasons. First, it scans fewer sequences. Second, the filtering step is also faster because k is much smaller than sequence length (usually some fixed constant, say, 6). Third, an index structure may be constructed in the k -dimensional space to speed up the filtering even further. Filtering may remove some of qualifying sequences resulting false-dismissals, because we cannot guarantee that the Euclidean distance in the $k-d$ space lower-bounds the time warping distance. This is the case even if we use the square root of the time warping distance, but the probability of false-dismissals is very low in practice, as we will see later.

4.2 Lower-bounding Technique

For two given sequences $\vec{x} = \langle x_1, \dots, x_m \rangle$ and $\vec{y} = \langle y_1, \dots, y_n \rangle$, let $\max_{\vec{x}}$ and $\max_{\vec{y}}$ denote the maximum values in \vec{x} and \vec{y} , respectively. $\min_{\vec{x}}$ and $\min_{\vec{y}}$ are defined similarly, but by the minimum values. A pair $\langle \max_{\vec{x}}, \min_{\vec{x}} \rangle$ defines a range within which a sequence \vec{x} can fluctuate. Without loss of generality, we assume $\max_{\vec{x}} \geq \max_{\vec{y}}$. In what follows, we make a series of observations to motivate a formal definition of a new distance function \mathcal{D}_b .

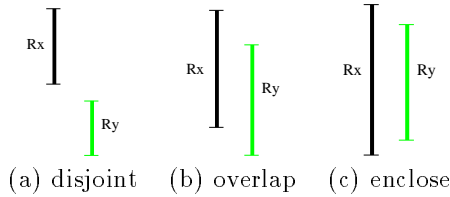


Figure 2: Possible arrangements of $R_{\vec{x}}$ and $R_{\vec{y}}$.

First, it is rather straightforward to see that there are only three possible arrangement of ranges of the two sequences being compared as seen in Figure 2.

Observation 1 Given two ranges, $R_{\vec{x}} = \langle \max_{\vec{x}}, \min_{\vec{x}} \rangle$ and $R_{\vec{y}} = \langle \max_{\vec{y}}, \min_{\vec{y}} \rangle$, there are three possible arrangements of the ranges.

- (C1) $R_{\vec{x}}$ and $R_{\vec{y}}$ are disjoint ($\min_{\vec{x}} > \max_{\vec{y}}$).
- (C2) $R_{\vec{x}}$ and $R_{\vec{y}}$ overlap ($\min_{\vec{x}} \leq \max_{\vec{y}}$, $\min_{\vec{x}} \geq \min_{\vec{y}}$).
- (C3) $R_{\vec{x}}$ encloses $R_{\vec{y}}$ ($\min_{\vec{x}} < \min_{\vec{y}}$).

When two sequences have disjoint ranges, we can observe the following:

Observation 2 Given two sequences \vec{x} and \vec{y} , if $\min_{\vec{x}} \geq \max_{\vec{y}}$, then $\sum_i |\min_{\vec{x}} - y_i| \leq \mathcal{D}_{\text{warp}}(\vec{x}, \vec{y})$, and $\sum_i |x_i - \max_{\vec{y}}| \leq \mathcal{D}_{\text{warp}}(\vec{x}, \vec{y})$.

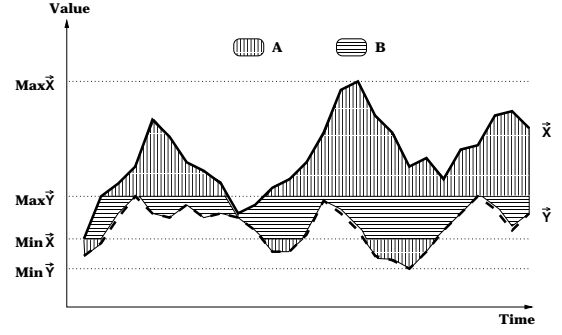


Figure 3: Illustration of the intuitive idea behind \mathcal{D}_b .

The explanation is that each y_i (respectively x_i) has to be counted in the warping distance eventually, and the least contribution it can make to the total distance is if it is compared against $\min_{\vec{x}}$ (respectively $\max_{\vec{y}}$).

Now consider two time sequences with overlapping ranges, such as \vec{x} (solid) and \vec{y} (dashed) in Figure 3. The shaded region between the two sequences is separated into two disjoint parts A and B. A is the shaded region above $\max_{\vec{y}}$ and below $\min_{\vec{x}}$, and B lies in between.

$\mathcal{D}_{\text{warp}}(\vec{x}, \vec{y})$ is just $\text{area}(A) + \text{area}(B)$ after time warping. Time warping attempts to minimize this sum by reducing $\text{area}(B)$, but it cannot reduce $\text{area}(A)$ since stuttering increases $\text{area}(A)$ for elements either below $\min_{\vec{x}}$ or above $\max_{\vec{y}}$, but has no effect on $\text{area}(A)$ for other elements. Suppose A' and B' denote A and B after time warping, respectively. Then we make the following observation.

Observation 3 $\text{area}(A) \leq \text{area}(A') \leq \text{area}(A') + \text{area}(B') = \mathcal{D}_{\text{warp}}(\vec{x}, \vec{y})$

We can make a similar observation when the range of \vec{x} encloses that of \vec{y} . See Figure 1 for instance.

From these observations, we are now ready to give a formal definition of our new distance function $\mathcal{D}_b()$ as follows:

Definition 2 (A new distance function \mathcal{D}_b)

$$\mathcal{D}_b(\vec{x}, \vec{y}) = \begin{cases} \max(\sum_i |x_i - \max_{\vec{y}}|, \sum_j |y_j - \min_{\vec{x}}|), & \text{(C1)} \\ \sum_i \phi(x_i, \max_{\vec{y}}) + \sum_j \phi(\min_{\vec{x}}, y_j), & \text{(C2)} \\ \sum_i \phi(x_i, \max_{\vec{y}}) + \sum_i \phi(\min_{\vec{y}}, x_i), & \text{(C3)} \end{cases}$$

$$\phi(a, b) = \begin{cases} |a - b|, & a > b \\ 0, & \text{otherwise} \end{cases}$$

where the conditions C1, C2, and C3 correspond to those in Observation 1. Note that both min and max values of a sequence can be calculated when the sequence is registered into a database by scanning once and can be stored with the sequence for future uses. Moreover, the arrangement of ranges of two sequences can be determined in constant time by simple comparisons. Finally, the definition of \mathcal{D}_b requires just one scan of each sequence, thus we can calculate the \mathcal{D}_b distance between two sequences in linear time in the

```

algorithm lower-bound-search
  R := {};
  /* filtering step */
  Given  $\vec{q}$ , foreach sequence  $\vec{s}_i$ 
  in the database S,
    if ( $\mathcal{D}_b(\vec{q}, \vec{s}_i) \leq \epsilon$ ),
      then add  $i$  to R;
  /* post-processing step */
  Foreach  $i$  in R,
    if ( $\mathcal{D}_{\text{warp}}(\vec{q}, \vec{s}_i) > \epsilon$ ),
      then remove  $i$  from R;
  Report R;
end algorithm

```

Algorithm 2: ϵ -Range Query based on \mathcal{D}_b .

length of sequences. This may result in a great improvement unless \mathcal{D}_b underestimates $\mathcal{D}_{\text{warp}}$ too much. We will verify our claim by experiments.

We claim that \mathcal{D}_b uniformly lower-bounds $\mathcal{D}_{\text{warp}}$ for any two sequences \vec{x} and \vec{y} .

Theorem 1 (Lower-bounding) *For any two sequences $\vec{x} = \langle x_1, \dots, x_m \rangle$ and $\vec{y} = \langle y_1, \dots, y_n \rangle$,*

$$\mathcal{D}_b(\vec{x}, \vec{y}) \leq \mathcal{D}_{\text{warp}}(\vec{x}, \vec{y})$$

Proof: We can prove it by elaborating the previous observations. For the complete proof, readers are referred to [14].

As a direct consequence of Theorem 1, we obtain the following corollary.

Corollary 1 (No False-Dismissals) *For any two sequences $\vec{x} = \langle x_1, \dots, x_m \rangle$ and $\vec{y} = \langle y_1, \dots, y_n \rangle$, if $\mathcal{D}_{\text{warp}}(\vec{x}, \vec{y}) \leq \epsilon$, then $\mathcal{D}_b(\vec{x}, \vec{y}) \leq \epsilon$.*

Lower-bounding the actual distance with another distance is a condition that guarantees no false dismissals for range queries and nearest neighbor queries[7]. Algorithm 2 describes how range queries can be processed. Other types of queries can be handled similarly.

In the filtering step, irrelevant sequences are filtered out quickly because the \mathcal{D}_b distance can be computed fast (linear time on the sequence length). Some non-qualifying sequences may be included in the result of this step because \mathcal{D}_b only lower-bounds $\mathcal{D}_{\text{warp}}$. The non-qualifying sequences are removed in the post-processing step.

Note that the algorithm does not reduce the number of sequences to be scanned. Instead, the speed-up comes from faster distance calculation: in linear, rather than in quadratic, time. In many applications, the length of sequences can be very long and this improvement is very important.

4.3 Combining the Two Techniques

In Algorithm 1, we compare filtered sequences using the time warping distance directly. However, we

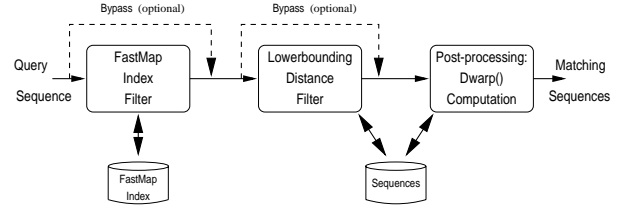


Figure 4: Proposed System Structure.

can use the lower-bounding distance before calculating time warping distance. It may require extra cost if the sequence is really a qualifying sequence, but may save much computation if it is not. This observation leads to a flexible multi-stage query processing system as shown in Figure 4, in which FastMap and \mathcal{D}_b serve as a primary and a secondary filter respectively.

It consists of three stages and they are connected in a pipelined manner. The input to each stage is a list of sequence IDs in the database of concern and the query sequence and output is a list of IDs of qualifying sequences at a stage. The first stage filters out irrelevant sequences using FastMap index only. Filtering at this stage reduces both I/O cost and CPU cost. Sequences that pass through the first filtering stage are compared with the query sequence by $\mathcal{D}_b()$ at the next stage. Finally, the post-processing stage selects only sequences that really match the query sequence. One of two filtering stages or both can be bypassed depending on the desirable time/recall trade-off.

5 Experimental Results

To show the effectiveness of our proposed methods, we performed experiments on real sequences (human electrocardiograms(ECG); and daily stock price data), as well as artificially generated sequences using sinusoids. Each query searched for sequences in the database matching a given query sequence (itself randomly chosen from the database) within a specified distance after warping. We compared the proposed methods and *sequential scanning* method in terms of average response time and average recall. All methods were implemented in C on a Pentium(100MHz) PC with 32 MB of memory and a 2GB Seagate SCSI disk(10msec average seek time), running FreeBSD(BSD4.4Lite-based). We measured the wall-clock time on this dedicated system.

We designed the experiments to answer the following questions:

- Which of the proposed techniques and their variant shows the best performance in terms of both response time and false dismissals?
- How well does each method scale as sequence length or database size grows?

The experimental parameters and their definitions are summarized in Table 2.

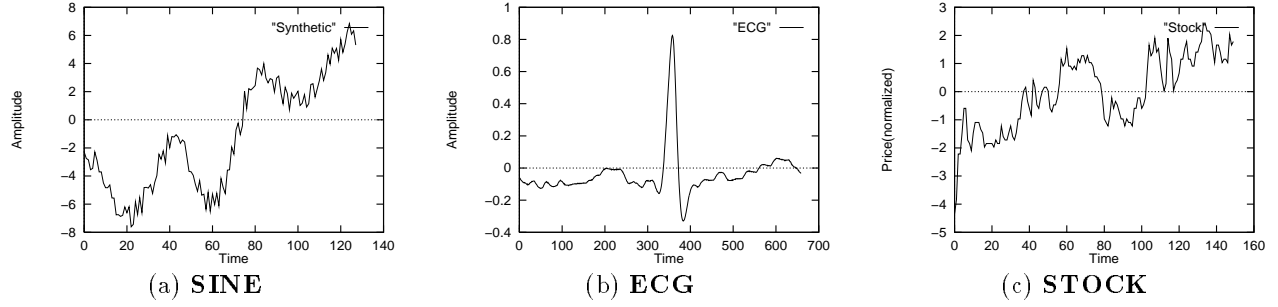


Figure 5: Sample Time Sequences.

Parameter	Definition
N	number of sequences in a database
L	(average) length of sequences
k	dimensionality of target space (= 6)

Table 2: Experimental Parameters and Definitions.

Dataset	Database Sz. (N)	Sequence Len. (L)
SINE	400	128
ECG	406	740
STOCK	640	150

Table 3: Experimental Parameter Settings.

5.1 Experimental Settings

For the experiment, we prepared three datasets. Samples of these time sequences are plotted in Figure 5.

- **SINE**: A dataset of synthetic time sequences. They were generated using sine curves as follows.

$$s(t) = \sum_{i=1}^S A_i \sin(f_i \cdot t + p_i) + \epsilon_i$$

where S is the number of sinusoids. A_i, f_i, p_i denote amplitude, frequency and phase of i -th sinusoid, respectively, and they were chosen randomly within some ranges. ϵ_i is a small white noise term. 400 sequences were generated, 100 for each $S = 2, 3, 4, 5$. Each sequence has length = 128.

- **ECG**: 406 sequences of human electrocardiogram (ECG) data. Their lengths vary from 640 to 840.
- **STOCK**: Stock price time sequences were generated by extracting 150 most recent (as of 6/5/96) daily high values from 640 stocks. These time sequences were normalized by subtracting the average, as was done in [5, 2].

Before building a FastMap index, we must determine the dimensionality k of the target space. In experiments not reported here for brevity, we observed that $k = 6$ was a good choice for all our the datasets. Other parameters are summarized in Table 3.

Four algorithms were compared:

- **Naive**: The straightforward method, bypassing both filters in Figure 4.
- **FM**: Algorithm 1, bypassing the lower-bounding distance filter in Figure 4.

- **LB**: Algorithm 2, bypassing the FastMap index filter in Figure 4.
- **FM+LB**: The proposed “combined” method, which enables both filters in Figure 4.

To measure how many false-dismissals are introduced by FastMap, we use the “recall” concept from Information Retrieval[10].

Definition 3 Recall is defined as follows:

$$\text{recall} \equiv \frac{\text{retrieved and relevant}}{\text{relevant}}$$

The (ideal) recall value of 1.0 means there are no false-dismissals, while a recall value of 0.0 means that no relevant objects are retrieved. Since we used two techniques, **Naive** and **LB**, that introduced no false-dismissals, the number of relevant objects for each query was known exactly.

5.2 Average Response Time and Recall

To compare the various proposed method, we performed range queries over 7 randomly selected query objects and calculated the average response time and recall. Search tolerances were chosen such that average number of matching sequences be approximately 1 (best match case) at the minimum tolerance and 5% of the database size at the maximum tolerance. Then, we compared the basic techniques (**LB**, **FM**) with the straightforward method (**Naive**). The results are shown in Figure 6. For all methods, response time grows as search tolerance. We observe that search tolerance has little effect on recall.

In response time, **FM** was the fastest of all methods. **LB** was comparable with **FM** in **STOCK**. In recall, the value of **LB** was always 1 as we expected. For **FM**, the value was 1 except for one case in **ECG** (≈ 0.964). Thus, we can conclude that all proposed

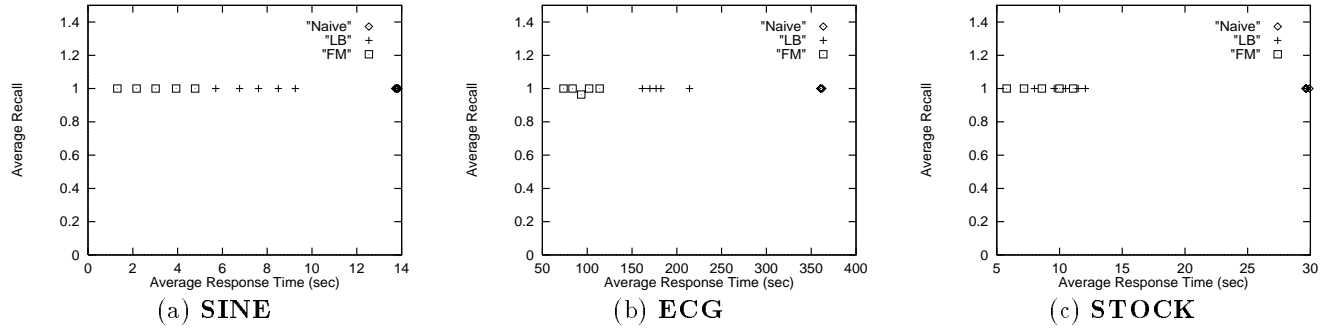


Figure 6: Comparison of **LB**(+) and **FM**(□) against **Naive**(◇). Recall vs. response time.

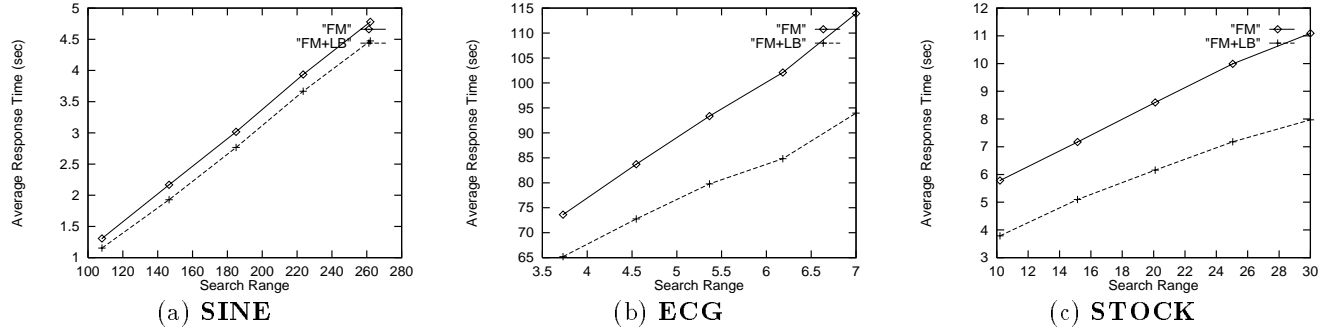


Figure 7: Comparison of **FM**(◇) against **FM+LB**(+). Response time vs. tolerance.

methods outperformed straightforward method in response time with little compromise in recall.

Next, we compared **FM** with **FM+LB**. Since the latter does not introduce any more false-dismissals than the former, we only compared them in terms of average response time. Figure 7 shows the result of this comparison. In all cases, the combined technique performed consistently faster than its basic counterpart.

Finally, we summarize the speed-up by all proposed techniques over **Naive** method at the minimum and maximum search tolerances in Table 4. The values report the ratio of the response time (**Naive** over the respective competitor). Notice that our proposed method achieves up to almost an order of magnitude (7.8 times) better response time, for real datasets (**STOCK**), and over an order of magnitude (12 times), for the synthetic **SINE** dataset. There is more speed-up for less selective range queries since fewer items are included in the answer set.

5.3 Scalability Test

In this section, we present the scalability test results on **FM+LB** method. Only **FM+LB** was chosen among proposed techniques, because it was clear in the previous section that it is the most promising method. Tests were performed in two ways. First, we generated extra synthetic datasets with varying lengths in the same way as previously and then performed range queries with a search tolerance so that as many as 5% of sequences in each dataset be retrieved.

Next, we generated 800 sequences of length 32 and ran range queries with a fixed search tolerance, over 200, 400, 600, and 800 sequences from this dataset. As we can see in Figure 8, the proposed technique scales up smoothly with both database size and sequence length, with increasing performance gap over the **Naive** method.

Remarkably, in all cases, the recall value of the proposed method was 1, indicating no false dismissals.

6 Conclusions

We focused on the fast similarity search on a large collection of time sequences, when the dissimilarity function is the “time-warping” distance [8], often used in audio and biological time sequences. The major contribution of this work is the idea to trade-off a tiny amount of “recall” (typically zero) to achieve significant speed-up (up to 7.8-time, on real data) We proposed and combined two methods:

- FastMap on the square-root of the time-warping distance, to map sequences to points, and,
- a lower-bounding, linear distance function, to accelerate the post-processing.

Minor contributions include:

- introduction of the time-warping distance to database audience, along with pointers to the related speech processing literature.
- implementation of proposed methods and experimental results on real and synthetic datasets

Method	Speed-up at Min Tolerance			Speed-up at Max Tolerance		
	SINE	ECG	STOCK	SINE	ECG	STOCK
LB	2.43	2.24	3.71	1.49	1.68	2.46
FM	10.59	4.92	5.31	2.88	3.17	2.67
FM+LB	12.01	5.56	7.82	3.08	3.84	3.71

Table 4: Speed-up by Proposed Techniques: ratio of response time of the **Naive** method over each competitor.

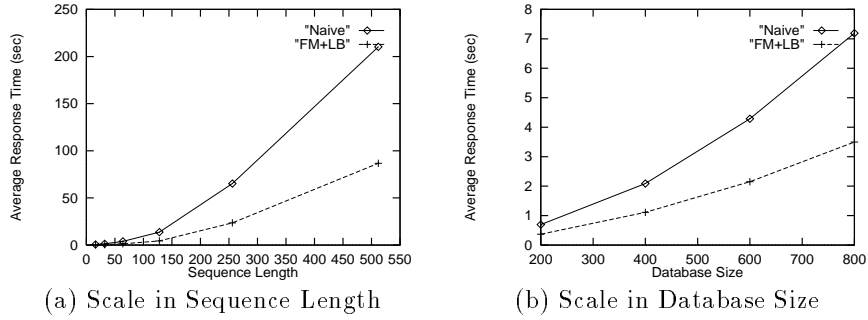


Figure 8: Scalability of **FM+LB**(+) and **Naive**(◇): response time vs. (a) sequence length (b) database size.

For future research, we plan to extend the proposed methods for time sequences of non-scalar values such as (feature) vectors.

References

- [1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *Proc. of the FODO Conf.*, Evanston, IL, October 1993.
- [2] Rakesh Agrawal, King-Ip Lin, Harpreet S. Sawhney, and Kyuseok Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series database. In *Proc. of the VLDB Conf.*, Zürich, Switzerland, 1995.
- [3] Christos Faloutsos and King-Ip Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. of the ACM SIGMOD Conf.*, San Jose, CA, June 1995.
- [4] Christos Faloutsos, M. Ranganathan, and Yan-nis Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. of the ACM SIGMOD Conf.*, May 1994.
- [5] Dina Q. Goldin and Paris C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proc. of Constraint Programming 95*, Marseilles, France, September 1995.
- [6] H. V. Jagadish, Alberto O. Mendelzon, and Tova Milo. Similarity-based queries. In *Proc. of the ACM PODS Conf.*, San Jose, CA, June 1995.
- [7] Flip Korn, Nicolaos Sidropoulos, and Christos Faloutsos. Fast nearest neighbor search in medical image databases. In *Proc. of the VLDB Conf.*, pages 215–226, Bombay, India, September 1996.
- [8] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.
- [9] Davood Rafiei and Alberto Mendelzon. Similarity-based queries for time series data. In *Proc. of the ACM SIGMOD Conf.*, Tucson, AZ, May 1997.
- [10] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
- [11] David Sankoff and Joseph B. Kruskal. *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparisons*. Addison-Wesley, 1983.
- [12] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. Sequence query processing. In *Proc. of the ACM SIGMOD Conf.*, pages 430–441, Minneapolis, MN, May 1994.
- [13] Dimitris Vassiliadis. The input-state space approach to the prediction of auroral geomagnetic activity from solar wind variables. *Int. Workshop on Applications of Artificial Intelligence in Solar Terrestrial Physics*, September 1993.
- [14] Byoung-Kee Yi, H.V. Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. Technical Report CS-TR-3837, Univ. of Maryland, September 1997. Also available as ISR TR 97-77 from “<http://www.isr.umd.edu>”.