Efficient Retrieval of Similar Time Sequences Under Time Warping

*Byoung-Kee Yi** *H.V. Jagadish*† *Christos Faloutsos**‡

October 9, 1997

Abstract

Fast similarity searching in large time-sequence databases has attracted a lot of research interest [1, 5, 2, 6, 3, 10]. All of them use the Euclidean distance ($\mathcal{L}_2$), or some variation of $\mathcal{L}_p$ metrics. $\mathcal{L}_p$ metrics lead to efficient indexing, thanks to feature extraction (*e.g.*, by keeping the first few DFT coefficients) and subsequent use of fast spatial access methods for the points in feature space.

In this work we examine a popular, field-tested dissimilarity function, the "time warping" distance function which permits local accelerations and decelerations in the rate of the signals or sequences. This function is natural and suitable for several applications, like matching of voice, audio and medical signals (*e.g.*, electrocardiograms) However, from the indexing viewpoint it presents two major challenges: (a) it does not lead to any natural "features", precluding the use of spatial access methods (b) it is quadratic ($O(len_1 * len_2)$) on the length of the sequences involved.

Here we show how to overcome both problems: for the former, we propose using a modification of the so- called "FastMap", to map sequences into points, trading off a tiny amount of "recall" (typically zero) for large gains in speed. For the latter, we provide 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 consistently outperformed the straightforward sequential scanning on both real and synthetic datasets and achieved up to 7.8-time speed-up (780%).

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.

*CS Dept., Univ. of Maryland, College Park, MD. {kee,christos}@cs.umd.edu

†AT&T Labs, Murray Hill, NJ. jag@research.att.com

‡This research was partially supported by Inst. for Systems Research, Univ. of Maryland, and by the NSF under Grants No. EEC-94-02384, IRI-9205273 and IRI-9625428.

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, 5]; scientific databases ("find times in the past that had similar solar magnetic wind patterns with the ones today" [14]).

In the area of speech recognition, this problem has been studied extensively, and is called the "(dynamic) 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 [4]. The idea here is to make use of the given distance measures to map sequences into points in k-d space, and to finally build an index structure. The other technique we propose defines a new distance function which uniformly underestimates the original distance function. This function can be computed much faster than the original distance so that it can be used as a filter to help us discard quickly non-qualifying sequences.

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 Works

Similarity-based matching of time sequences has attracted a lot of attention in the signal processing area, and specifically in speech processing. However, they assumed a small dataset (*e.g.*, a few tens of phonemes) and were more concerned with the 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 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 [5] we generalized the approach for subsequence matching.

Follow-up work by Goldin and Kanellakis [6] 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.

Rafiei and Mendelzon [10] 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 [7], we *et al* develop 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* [13] 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, 5] 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 sampled between the same time intervals, and samples are real numbers, *e.g.*, $\langle x_1, \ldots, x_n \rangle$ such that $x_i = f(i \times \Delta t)$ for some (unknown) real-valued function f. We will denote a sequence $\langle x_1, \ldots, x_n \rangle$ as $\vec{x}$. Table 3 gives a list of symbols used in the rest of the paper.

We consider the $\mathcal{L}_p$ metric family of distance functions. For two sequences $\vec{x} = \langle x_1, \ldots, x_n \rangle$, $\vec{y} = \langle y_1, \ldots y_n \rangle$ this distance is defined by the formula

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

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 $\vec{x} = \langle x_1, \ldots, x_n \rangle$, let $Head(\vec{x})$ denote x_1 and $Rest(\vec{x})$ denote $\langle x_2, \ldots, x_n \rangle$. Then, we want to allow the stuttering transformation on a sequence, possibly with a penalty ("cost"):

- $stutter_i(\vec{x})$: repeats x_i and shifts the elements to the right.

Following [9], for any non-null sequences $\vec{x}$ and $\vec{y}$, the (dynamic) time warping distance is defined as follows.

Symbol	Definition
$\mathcal{D}_p$	$\mathcal{L}_p$-based distance function
$\mathcal{D}_{base}$	base distance function, *e.g.*, $\mathcal{D}_1$ 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 $\vec{x}$
$\lvert\vec{x}\rvert$	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

Definition 1 *The time warping distance between two sequences is defined as:*

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

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.

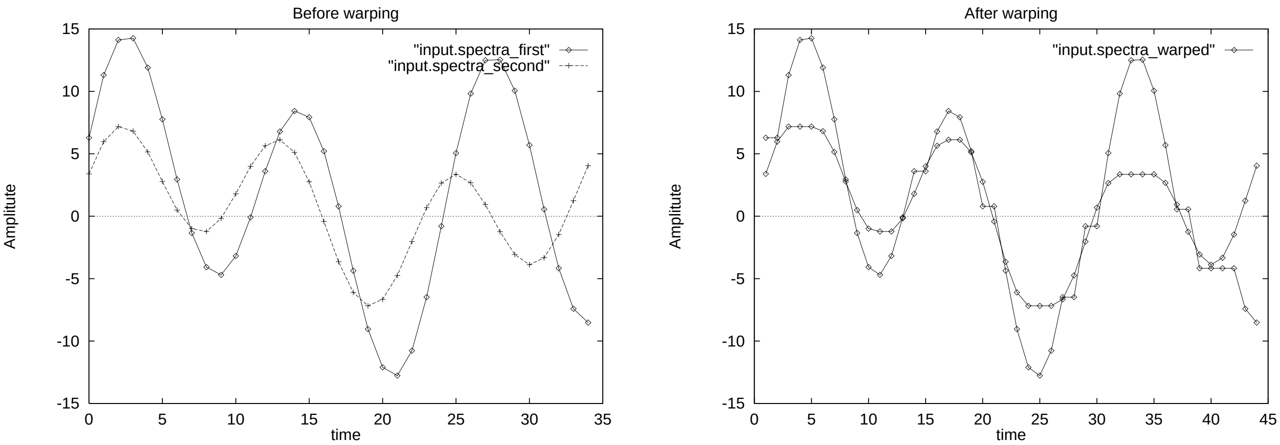
Defined as a recurrence, the time-warping distance can be computed by a dynamic programming algorithm (see Appendix A or [9]) whose complexity is $O(|\vec{x}| \times |\vec{y}|)$. See [12] for more details and other variants of the basic algorithm. Figure 1 shows two time sequences, before and after the 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.

Proposition 1 $\mathcal{D}_{warp}()$ *does not satisfy 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

$$\mathcal{D}_{warp}(\vec{x},\ \vec{z}) = 5 > \mathcal{D}_{warp}(\vec{x},\ \vec{y}) + \mathcal{D}_{warp}(\vec{y},\ \vec{z}) = 3 + 0 = 3$$

which completes the proof.



(a) before (b) after

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

This fact has significant implication on the method we can use for indexing: *Any* indexing technique which assumes the triangle inequality implicitly or explicitly, can not avoid producing false dismissals. This is a very strict requirement: all the spatial access methods, as well as all the methods that use distance/metric/vantage-point trees, can not avoid false dismissals. The only method that guarantees no false dismissals is sequential scanning, which will be prohibitive for a large collection of long sequences, because of the the quadratic nature of Algorithm 3.

To resolve the issue, we propose a method that trades-off a tiny percentage of false dismissals, for significant speed-up. Namely, our goal is to provide an efficient retrieval technique, while keeping the false dismissals as few as possible.

4 Proposed Techniques

Suppose there is a database containing many time sequences of arbitrary length and that a user wants to find all sequences similar to a certain query sequence, that is, all sequences within ϵ units of time-warping distance. This type of query is effectively a range query.

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 a few false dismissals. We will also discuss how to reduce false dismissals.
- To use a new distance function which 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.

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" [4]. 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 every other method (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 fastmap-range-search
   R := {}; /* response set */
   /* filtering step */
   Given q⃗, foreach sequence s⃗_i in the database S,
      if (D_2(F(q⃗), F(s⃗_i)) ≤ ε), then add i to R;
   /* post-processing step */
   Foreach i in R,
      if (D_warp(q⃗, s⃗_i) > ε), then remove i from R;
   Report R;
end algorithm
```

Algorithm 1: Pseudo-code of ϵ-Range Query using on FastMap

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 two reasons. First, it scans fewer of sequences. Second, the filtering step is also faster because k is much smaller than sequence length (usually some fixed constant, say, 6). Filtering may remove some of qualifying sequences resulting false-dismissals, because we can not 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 $\vec{x} = \langle x_1, \ldots, x_m \rangle$ and $\vec{y} = \langle y_1, \ldots, y_n \rangle$, 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 the range within which the sequence $\vec{x}$ can fluctuate. Without loss of generality, we assume $max(\vec{x}) \geq max(\vec{y})$.[1] The proposed method is motivated by the following observation:

Observation 1 $|max(\vec{x}) - max(\vec{y})| \leq \mathcal{D}_{warp}(\vec{x}, \vec{y})$.

Checking its validity is rather straightforward. Since $max(\vec{x})$ should match at least an element of $\vec{y}$, say y_i, and we assumed $max(\vec{x}) \geq max(\vec{y})$,

$$|max(\vec{x}) - max(\vec{y})| \leq |max(\vec{x}) - y_i| \leq \mathcal{D}_{warp}(\vec{x}, \vec{y}).$$

The consequence of this observation is the absolute difference in the maximum values can serve as a distance that lower-bounds the time warping distance. While this is true, however, it may not be very useful because it may underestimate too much. Thus, our goal is to find a tighter lower-bound.

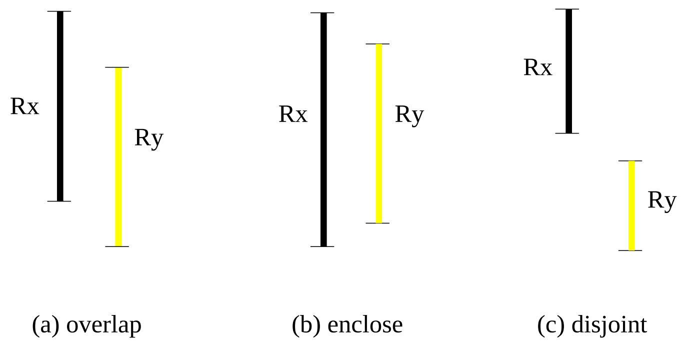


Figure 2: Possible arrangements of $R_{\vec{x}}$ and $R_{\vec{y}}$

We consider the possible arrangement of ranges of two sequences being compared.

[1]Otherwise, we can switch their roles.

Observation 2 *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(see Figure 2).*

1. $R_{\vec{x}}$ *and* $R_{\vec{y}}$ *overlap* $(min(\vec{x}) \leq max(\vec{y}), min(\vec{x}) \geq min(\vec{y}))$.
2. $R_{\vec{x}}$ *encloses* $R_{\vec{y}}$ $(min(\vec{x}) < min(\vec{y}))$.
3. $R_{\vec{x}}$ *and* $R_{\vec{y}}$ *are disjoint* $(min(\vec{x}) > max(\vec{y}))$.

From these observations, we define a new distance function $\mathcal{D}_{lb}()$ as follows:

Definition 2 (A new distance function $\mathcal{D}_{lb}$)

$$\mathcal{D}_{lb}(\vec{x},\vec{y}) = \begin{cases} \sum_{x_i > max(\vec{y})} |x_i - max(\vec{y})| + \sum_{y_j < min(\vec{x})} |y_j - min(\vec{x})| & \text{if } R_{\vec{x}} \text{ and } R_{\vec{y}} \text{ overlap} \\ \sum_{x_i > max(\vec{y})} |x_i - max(\vec{y})| + \sum_{x_i < min(\vec{y})} |x_i - min(\vec{y})| & \text{if } R_{\vec{x}} \text{ encloses } R_{\vec{y}} \\ max(\sum_{i=1}^{m} |x_i - max(\vec{y})|, \sum_{j=1}^{n} |y_j - min(\vec{x})|) & \text{if } R_{\vec{x}} \text{ and } R_{\vec{y}} \text{ are disjoint} \end{cases}$$

Note that both minimum and maximum 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 $\mathcal{D}_{lb}$ requires just one scan of each sequence, thus we can calculate the $\mathcal{D}_{lb}$ distance between two sequences in linear time in the length of sequences. This may result in a great improvement unless $\mathcal{D}_{lb}$ underestimates $\mathcal{D}_{warp}$ too much. We will verify our claim by experiments.

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

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

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

Proof: See Appendix B.

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

Corollary 1 (No False-Dismissals) *For any two sequences* $\vec{x} = \langle x_1, \ldots, x_m\rangle$ *and* $\vec{y} = \langle y_1, \ldots, y_n\rangle$,

$$\text{if } \mathcal{D}_{warp}(\vec{x},\vec{y}) \leq \epsilon, \text{ then } \mathcal{D}_{lb}(\vec{x},\vec{y}) \leq \epsilon.$$

As we have shown in a previous paper[5], lower-bounding the actual distance with another distance is a condition that guarantees no false dismissals for range queries and nearest neighbor queries[8].

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 $\mathcal{D}_{lb}$ distance can be computed fast (linear time on the dimensionality k, typically $k \leq 10$). Some non-qualifying sequences may be included in the result of this step because $\mathcal{D}_{lb}$ lower-bounds our object distance $\mathcal{D}_{warp}$. However, those non-qualifying sequences are removed in the post-processing step.

```
algorithm lower-bound-search
   R := {};
   /* filtering step */
   Given q⃗, foreach sequence s⃗_i in the database S,
      if (D_lb(q⃗, s⃗_i) ≤ ε), then add i to R;
   /* post-processing step */
   Foreach i in R,
      if (D_warp(q⃗, s⃗_i) > ε), then remove i from R;
   Report R;
end algorithm
```

Algorithm 2: ϵ-Range Query based on $\mathcal{D}_{lb}$

Note that the algorithm does not reduce the number of sequences to be scanned. Instead, the speed-up comes from faster distance calculation. But, in many applications, the length of sequences can be very long and quadratic-time distance calculation should be avoid as often as possible. This fact justifies the effectiveness of the algorithm.

4.3 Combining the Two Techniques

A careful consideration of the two techniques proposed in the previous sections can lead more efficient algorithm. In Algorithm 1, we compare filtered sequences using the time warping distance only. However, we 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 great amount of computational cost otherwise. This observation leads to a flexible multi-stage query processing system as shown in Figure 3, in which FastMap and $\mathcal{D}_{lb}$ serve as a primary and a secondary filter respectively.

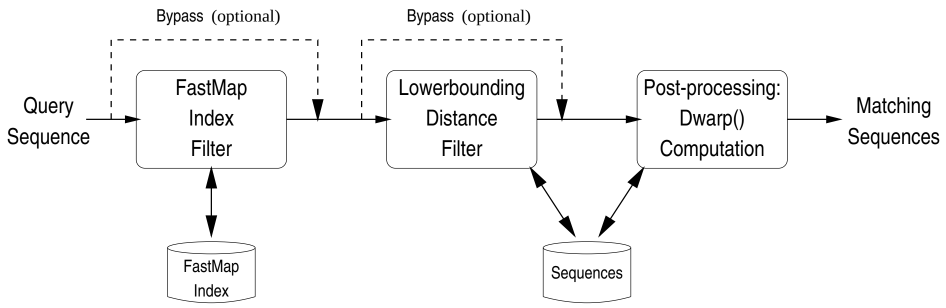


Figure 3: Proposed System Structure

It consists of three stages and they are connected in a pipelined manner. The input to each stage is a list of sequence ID's in the database of concern and the query seqence and output is a list of ID's 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. Those sequences that pass through the first filtering stage are compared with the query sequence by $\mathcal{D}_{lb}()$ at the next stage. Finally, the post-processing stage selects only those sequences which 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 time sequences (human electrocardiograms(ECG); and daily stock price data), as well as artificially generated sequences using sinusoids. Range queries with varying query objects and search ranges (= tolerances) were performed on these sequences. We compared 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 each method scales as sequence length or dabase size grows?

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

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

Table 2: Summary of Experimental Parameters and Definitions

5.1 Experimental Settings

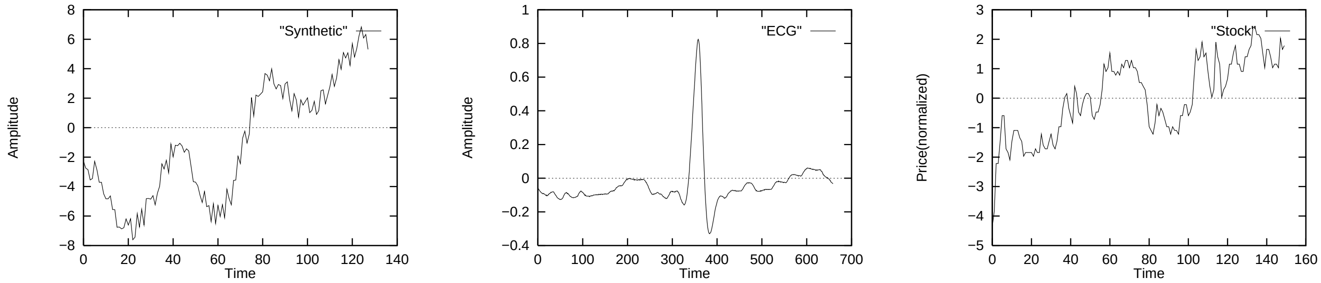
For the experiment, we prepared three datasets. Samples of these time sequences are plotted in Figure 4.

- **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 [6, 2].



(a) **SINE** (b) **ECG** (c) **STOCK**

Figure 4: Sample Time Sequences

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.

Dataset	Database Size (N)	Sequence Length (L)
SINE	400	128
ECG	406	740
STOCK	640	150

Table 3: Experimental Parameter Settings

As the range query methods, 4 algorithms were compared. These include,

- **Naive**: The straightforward method, bypassing both filters in Figure 3.
- **FM**: Algorithm 1, bypassing the lower-bounding distance filter in Figure 3.
- **LB**: Algorithm 2, bypassing the FastMap index filter in Figure 3.
- **FM+LB**: The proposed "combined" method, which enables both filters in Figure 3.

To measure how many false-dismissals are introduced by FastMap, we use the "recall" concept from Information Retrieval[11].

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.

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 ranges were chosen such that average number of matching sequences be approximately 1 (best match case) at the minimum range and 5% of the database size at the maximum range. Then, we compared the basic techniques(**LB**, **FM**) with the straightforward method(**Naive**). The results are shown in Figure 5. For all methods, response time grows as search range. We observe that search range 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** ($\approx$ 0.964). Thus, we can conclude that all proposed methods outperformed straightforward method in response time with little compromise in recall.

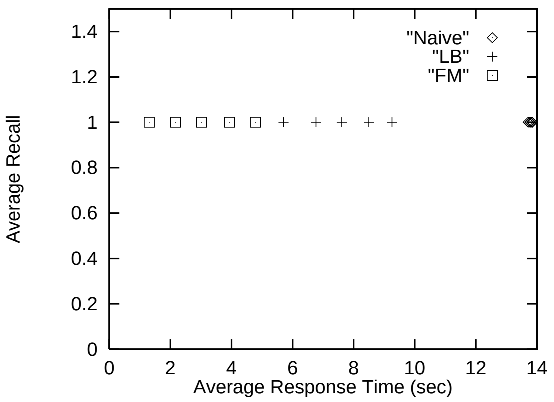
Next, we compared **FM** with **FM+LB**. Since the latter does not introduce any more false-dismissals than the former, we only compared them in terms of average response time. Figure 6 shows the result of this comparison. In all cases, the combined technique performed consistently faster than its basic counterpart.

Finally, we summarize the speed-up by all proposed techniques over **Naive** method at the minimum and maximum search ranges 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.

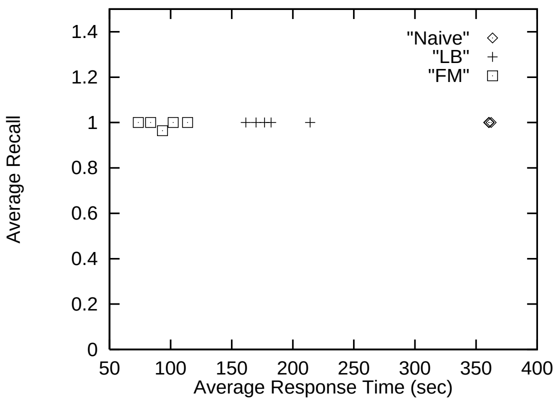
5.3 Scalability Test

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 range so that as many as 5% of sequences in each dataset be retrieved. Figure 7(a) shows the result. We can see that proposed technique performs about 3 times faster than **Naive** method.

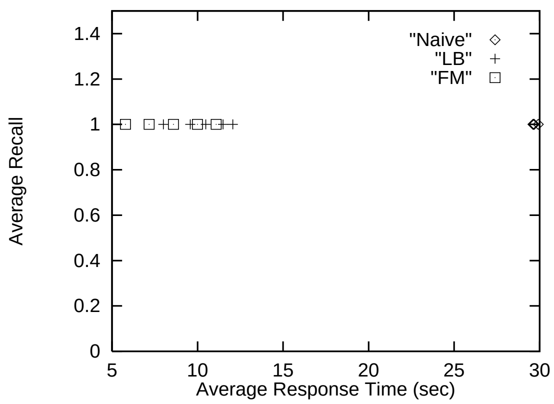
Next, we generated 800 sequences of length 32 and ran range queries with a fixed search range, over 200, 400, 600, and 800 sequences from this dataset. As we can see in Figure 7, the proposed technique scales up smoothly with the database size, with increasing performance gap over the **Naive** method.



(a) **SINE**

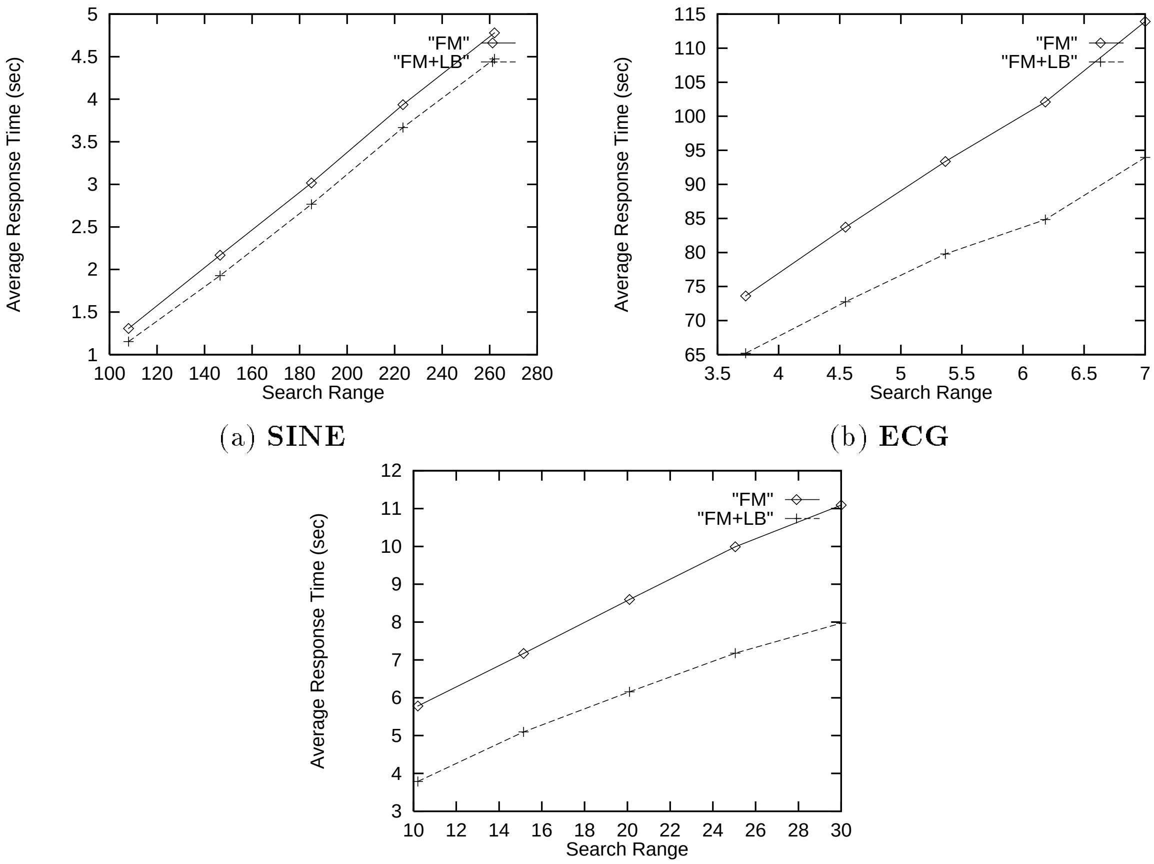


(b) **ECG**



(c) **STOCK**

Figure 5: Comparison of the Basic Techniques (**LB** with crosses; **FM** with squares) against the Naive Method (**Naive** with diamonds).



(a) **SINE**

(b) **ECG**

(c) **STOCK**

Figure 6: Comparison of FastMap (**FM** with diamonds) against the combined technique (**FM+LB** with crosses). Response time versus tolerance.

Method	Speed-up at Min Range	Speed-up at Max Range
LB	2.43	1.49
FM	10.59	2.88
FM+LB	12.01	3.08

(a) **SINE**

Method	Speed-up at Min Range	Speed-up at Max Range
LB	2.24	1.68
FM	4.92	3.17
FM+LB	5.56	3.84

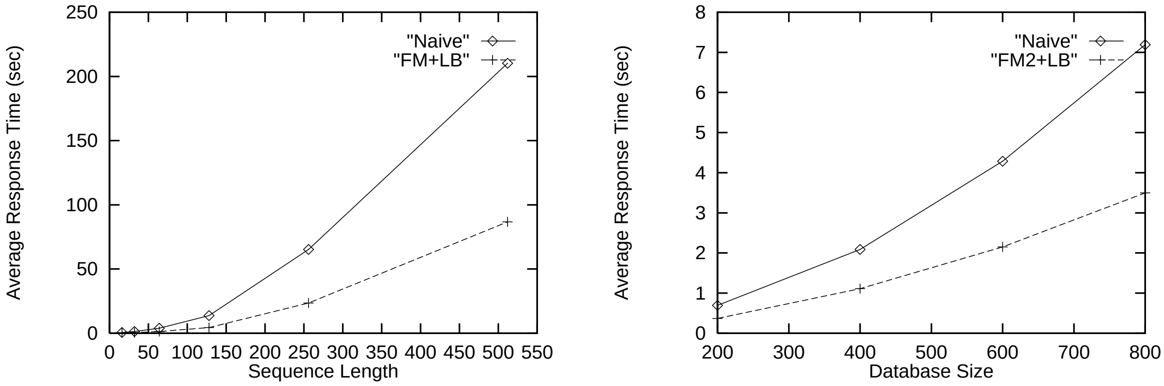
(b) **ECG**

Method	Speed-up at Min Range	Speed-up at Max Range
LB	3.71	2.46
FM	5.31	2.67
FM+LB	7.82	3.71

(c) **STOCK**

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

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



(a) Scale in Sequence Length (b) Scale in Database Size

Figure 7: Scalability of **FM+LB**: response time versus (a) average sequence length (b) database size (**Naive** with diamonds; **FM+LB** with crosses).

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 [9], as it happens 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
- a lower-bounding, linear distance function, to accelerate the post-processing.

Minor contributions include:

- the 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

References

[1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *Proceedings of the FODO Conference*, Evansotn, IL, USA, 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 *Proceedings of the 21st VLDB Conference*, Zürich, Switzerland, 1995.

[3] C. Faloutsos, H. V. Jagadish, A. O. Mendelzon, and T. Milo. A signature technique for similarity-based queries. In *Proceedings of SEQUENCES97*, Salerno, Italy, June 1997.

[4] Christos Faloutsos and King-Ip Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the ACM SIGMOD Conference*, San Jose, CA, USA, June 1995.

[5] Christos Faloutsos, M. Ranganathan, and Yannis manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the ACM SIGMOD Conference*, May 1994.

[6] Dina Q. Goldin and Paris C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of Constraint Programming 95*, Marseilles, September 1995.

[7] H. V. Jagadish, Alberto O. Mendelzon, and Tova Milo. Similarity-based queries. In *Proceedings of the ACM PODS Conference*, San Jose, CA, USA, June 1995.

[8] Flip Korn, Nicolaos Sidropoulos, and Christos Faloutsos. Fast nearest neighbor search in medical image databases. In *Proceedings of VLDB Conference*, pages 215–226, Bombay, India, September 1996.

[9] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[10] Davood Rafiei and Alberto Mendelzon. Similarity-based queries for time series data. In *Proceedings of the ACM SIGMOD Conference*, Tucson, AZ, May 1997.

[11] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[12] David Sankoff and Joseph B. Kruskal. *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparisons.* Addison-Wesley Publishing Company, Inc., Reading, MA, 1983.

[13] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. Sequnce query processing. In *Proceedings of the ACM SIGMOD Conference*, pages 430–441, Minneapolis, MN, USA, May 1994.

[14] 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.

A Appendix: Time Warping Distance Algorithm

Here we give the pseudo-code to compute the time-warping distance between two sequences $\vec{x}$ and $\vec{y}$. The idea is to build an auxiliary matrix $M[i, j]$, which will store the cost of matching the first i samples of $\vec{x}$ with the first j samples of $\vec{y}$; thus, $M[m, n]$ is the desired cost (if m and n are the lengths of $\vec{x}$ and $\vec{y}$ respectively).

```
algorithm time-warping-distance(x⃗, y⃗)
   /* Initializations */
   m := |x⃗|; n := |y⃗|;
   M[0,0] := 0.0;
   M[1...m,0] := ∞;
   M[0,1...n] := ∞;
   /* Compute partial results */
   for 1 ≤ i ≤ m,
      for 1 ≤ j ≤ n,
         M[i,j] := D_base(x_i, y_j) + min{ M[i−1,j], M[i,j−1], M[i−1,j−1] };
   return M[m,n];
end algorithm
```

Algorithm 3: Time Warping Distance

B Appendix: Proof of Theorem 1

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

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

Proof: Let $w_{\vec{x}}(k)$ and $w_{\vec{y}}(k)$ be two warping functions whose domains are both $\{1, \cdots, M\}$ such that

$$\sum_{k=1}^{M} |x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| = \mathcal{D}_{warp}(\vec{x}, \vec{y}).$$

There must exist such warping functions, although they may not be unique. We will show that the theorem holds for each possible arrangement of ranges.

$R_{\vec{x}}$ and $R_{\vec{y}}$ overlap: For an arbitrary k, there are three possibilities.

case 1: $x_{w_{\vec{x}}(k)} > max(\vec{y})$ and $y_{w_{\vec{y}}(k)} < min(\vec{x})$

$$|x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| \geq |x_{w_{\vec{x}}(k)} - max(\vec{y})| + |y_{w_{\vec{y}}(k)} - min(\vec{x})|.$$

case 2: $x_{w_{\vec{x}}(k)} > max(\vec{y})$ and $y_{w_{\vec{y}}(k)} \geq min(\vec{x})$

$$|x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| \geq |x_{w_{\vec{x}}(k)} - max(\vec{y})|.$$

case 3: $x_{w_{\vec{x}}(k)} \leq max(\vec{y})$ and $y_{w_{\vec{y}}(k)} < min(\vec{x})$

$$|x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| \geq |y_{w_{\vec{y}}(k)} - min(\vec{x})|.$$

Other cases are either not possible, or irrelevant because none of $x_{w_{\vec{x}}(k)}$ and $y_{w_{\vec{y}}(k)}$ contributes to $\mathcal{D}_{lb}$. Note that these cases are mutually exclusive. Thus, adding up both sides of the above inequalities through all k proves the theorem.

$R_{\vec{x}}$ **encloses** $R_{\vec{y}}$**:** For an arbitrary k, there are only two possibilities.

case 1: $x_{w_{\vec{x}}(k)} > max(\vec{y})$

$$|x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| \geq |x_{w_{\vec{x}}(k)} - max(\vec{y})|.$$

case 2: $x_{w_{\vec{x}}(k)} < min(\vec{y})$

$$|x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| \geq |x_{w_{\vec{x}}(k)} - min(\vec{y})|.$$

Similarly, adding up both sides of the above inequalities through all k proves the theorem.

$R_{\vec{x}}$ **and** $R_{\vec{y}}$ **are disjoint:** For an arbitrary k,

$$x_{w_{\vec{x}}(k)} > max(\vec{y}) \quad \text{and} \quad y_{w_{\vec{y}}(k)} < min(\vec{x}).$$

Consequently we have,

$$\sum_{k=1}^{M} |x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| \geq \sum_{k=1}^{M} |x_{w_{\vec{x}}(k)} - max(\vec{y})|.$$

Also,

$$\sum_{k=1}^{M} |x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| \geq \sum_{k=1}^{M} |y_{w_{\vec{y}}(k)} - min(\vec{x})|.$$

Therefore,

$$\sum_{k=1}^{M} |x_{w_{\vec{x}}(k)} - y_{w_{\vec{y}}(k)}| \geq max(\sum_{k=1}^{M} |x_{w_{\vec{x}}(k)} - max(\vec{y})|, \sum_{k=1}^{M} |y_{w_{\vec{y}}(k)} - min(\vec{x})|).$$

From the above case analysis, the theorem holds. □

Contents