Fast Time Sequence Indexing for Arbitrary $\mathcal{L}_p$ Norms

Byoung-Kee Yi[*]
Dept. of Computer & Info. Science
New Jersey Institute of Technology
kee@cis.njit.edu

Christos Faloutsos[†]
Dept. of Computer Science
Carnegie Mellon University
christos@cs.cmu.edu

Abstract

Fast indexing in time sequence databases for similarity searching has attracted a lot of research recently. Most of the proposals, however, typically centered around the Euclidean distance and its derivatives. We examine the problem of multimodal similarity search in which users can choose the best one from multiple similarity models for their needs.

In this paper, we present a novel and fast indexing scheme for time sequences, when the distance function is any of arbitrary $\mathcal{L}_p$ norms ($p = 1, 2, \ldots, \infty$). One feature of the proposed method is that only one index structure is needed for all $\mathcal{L}_p$ norms including the popular Euclidean distance ($\mathcal{L}_2$ norm). Our scheme achieves significant speedups over the state of the art: extensive experiments on real and synthetic time sequences show that the proposed method is comparable to the best competitor for $\mathcal{L}_2$ and $\mathcal{L}_\infty$ norms, but significantly (up to 10 times) faster for $\mathcal{L}_1$ norm.

1 Introduction

Time sequences of real-values arise in many applications such as stock market, medicine/science, and multimedia.

[*]This work was partly done while this author was at the University of Maryland, College Park.

[†]This material is based upon work supported by the National Science Foundation under Grants No. IRI-9625428, DMS-9873442, IIS-9817496, and IIS-9910606, and by the Defense Advanced Research Projects Agency under Contract No. N66001-97-C-8517. Additional funding was provided by donations from NEC and Intel. 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, DARPA, or other funding parties.

Proceedings of the 26th VLDB Conference,
Cairo, Egypt, 2000.

Retrieval of these sequences is based on *'similarity'* as opposed to *exact equality*. For instance, a financial analyst may be interested in such queries as:

- *"Find all stocks whose prices moved similarly to that of a company over the last two months."*
- *"Find all companies which have similar patterns of revenue growth to that of another company for the last decade."*
- *"Find all currencies whose prices w.r.t. US Dollar have changed similarly to the price of gold for a specific period of time.*

Results of the queries can be used for further analysis of the market trends and/or key factors behind certain market events.

Similarity-based search in large collections of time sequences has attracted a lot of research recently in database community, including [1, 9, 11, 2, 19, 24], to name just a few. Main focus has been fast indexing techniques to improve performance when a particular similarity model is given. Typically, sequences of fixed length are mapped to points in an N-dimensional Euclidean space and, then, multi-dimensional access methods such as R-tree family [12, 21, 3] can be used for fast access of those points.

Since, however, time sequences are usually long, a straightforward application of the above approach suffers from performance degradation due to a phenomenon known as *'dimensionality curse.'* [4] To address the problem, several dimensionality-reduction techniques have been proposed. *Discrete Fourier Transform* (DFT) was the most popular and used in [1, 9, 11, 19] and, more recently, *Discrete Wavelet Transform* (DWT) was also proposed [15]. The basic idea is to approximate original time sequences with a few transform coefficients and, hence, map them into low-dimensional points. These methods guarantee that every qualifying sequence will be retrieved (*no false dismissals*). Some non-qualifying sequences may be retrieved, but can be removed in the post-processing stage. Other techniques include *piece-wise constant* approximation [7], and *FastMap* [8, 24].

Another issue in the area has been the choice of similarity models. *Euclidean* distance ($\mathcal{L}_2$ norm) was the most heavily used one [1, 9, 11, 19]. Linear correlation coeff. [15] is closely related to the normalized Euclidean distance [8]. We investigated the *Time Warping* distance in [24]. Infinity norm ($\mathcal{L}_\infty$) was proposed in [2] as part of a more complex similarity model. Other similarity models are also possible and have been proposed, but due to the space limitation we do not discuss them any further.

We note the following limitations in the previous approaches.

- **Multi-Modality Support:** No single model of similarity is suitable for every application. Sometimes several similarity models may be required for the same database of sequences, depending on different perspectives of different users. (Even a single user may want to have multiple models.) No previous work has proposed a single framework to support this *multi-modal* query processing for time sequences. To support it, we are forced to implement different techniques for different models into a DBMS, which is not efficient and only add complexity to the system, making it hard to build core DBMS components such as query optimizer, since different techniques may require different access methods and storage organizations *etc.*

- **Feature Extraction:** Proposed feature extraction (dimensionality reduction) methods are either (a) only suitable for a particular similarity model, or (b) have other shortcomings. For example, DFT as well as DWT has been shown very effective when the given distance function is Euclidean, but its effectiveness is questionable for other similarity models. FastMap may be used for a wider class of models, but it does not guarantee '*no false dismissal*.' Piece-wise constant approximation does not allow for indexing due to its irregularity.

In this paper, we address the above problems and propose a new similarity-based query processing scheme for time sequences. We focus on arbitrary $\mathcal{L}_p$ norms, since they have been widely used in real applications and can be used as basic building blocks for more complex similarity models as in [2].

We propose a new feature extraction method based on *segmented means*. We divide each time sequence into a fixed number, say 's', equal sized segments and take the mean of each segment to form a feature vector. It has a nice mathematical property such that we can decrease the given search range without affecting the correctness of the query results. Moreover, the proposed method provides a single unified framework in which,

- multiple similarity models are supported simultaneously,
- indexing for fast retrieval is supported, and,
- the same index structure can be re-used for different models.

Symbol	Definition
DFT	Discrete Fourier Transform
DWT	Discrete (Haar) Wavelet Transform
$\vec{x}$	a time sequence
x_i	the i-th value of $\vec{x}$ ($1 \leq i \leq L$)
$\lvert\vec{x}\rvert$	length of $\vec{x}$
s	number of segments
l	length of each segment (= $\lceil L/s \rceil$)
P_j^x	the j-th segment of $\vec{x}$ ($1 \leq j \leq s$ and $\lvert P_j^x\rvert = l$)
F_s^x	feature vector of $\vec{x}$
ϵ	search tolerance
w	sliding window size for the subsequence matching

Table 1: List of symbols

We will demonstrate the efficiency of the method via extensive experiments based on *whole-sequence* and *subsequence* matching queries against a stock price dataset as well as a synthetic dataset.

Organization of the paper In Section 2, we survey related work. In Section 3, we present our proposed method in detail as well as how to use existing techniques. Section 4 reports experimental results to compare the proposed method and the competitors. Finally, Section 5 discusses the key contributions of the paper. In Table 1, we list the symbols and their definitions that we use in the rest of the paper.

2 Related Work

Similarity-based matching of time sequences has been studied extensively in the signal processing area, and specifically in speech processing [18]. 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.

Performance is the main focus in the recent database work on sequence matching. They differ in what type of distance function is used and what type of matching they aim at. In [1], Agrawal *et al* examined the whole matching problem when the given dissimilarity function is the Euclidean distance, and suggest using the Discrete Fourier Transform (DFT). They argued that most of real signals need only a few DFT coefficients to approximate them. They proposed an indexing mechanism called *F-Index* which takes a few of the first coefficients and regards them as a point in the Euclidean space. Hence it makes possible to use any of readily available multidimensional access methods. The proposed method may allow a few false alarms which can be removed in the post-processing stage, but guarantees no false dismissals. In [9], authors generalized the approach for subsequence matching. Follow-up work by Goldin and Kanellakis [11] suggested that we normalize the sequences first, to allow for differences in level and scale. 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

Ref.	Dist.	Features	Matching	Index	Transformations
[1]	$\mathcal{L}_2$	DFT	whole	yes	none
[9]	$\mathcal{L}_2$	DFT	subseq	yes	none
[11]	$\mathcal{L}_2$	DFT	whole	yes	offset translation, amplitude scaling
[2]	$\mathcal{L}_\infty$ based	none	subseq	yes	offset translation, amplitude scaling, gaps allowed
[19]	$\mathcal{L}_2$	DFT	whole	yes	moving average, time scaling
[15]	Corr. Coeff.	DFT or DWT	subseq	no	offset translation, amplitude scaling
[7]	$\mathcal{L}_2$ variant	piece-wise constant	whole	no	regional add
[24]	Time Warping	FastMap	whole	yes	none
[13]	$\mathcal{L}_2$	PAA	whole	yes	weighting

Table 2: Comparison of different approaches

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 [19] 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 [15], authors proposed a hierarchical scanning method based on the linear correlation coefficient as a similarity measure. Faloutsos *et al* [7] proposed a generic framework for similar time sequences. It takes advantage of piece-wise constant approximations as signatures for fast comparison of sequences and allows for *regional add* transform. We proposed efficient techniques when the similarity measure is defined by the time warping distance [24]. Keogh and Pazzani [13] proposed a feature extraction method which is coincidently vey similar to ours, but their focus was on $\mathcal{L}_2$. These approaches are summarized in Table 2. We compared them in terms of the distance metrics, the features, the type of matching, the possibility of indexing and the allowed transformations.

3 Indexing Time Sequences for $\mathcal{L}_p$ Norms

Different dissimilarity measures have been discussed in the literature. Among others, $\mathcal{L}_p$ norm is the most popular class of dissimilarity measures and defined as follows:

$$\mathcal{L}_p(\vec{x}-\vec{y}) = \left(\sum_{i=1}^{L} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

It is called the *city-block* or the *Manhattan* norm when $p = 1$, and the *Euclidean* norm when $p = 2$. In the extreme case when $p = \infty$, it is called the *maximum* norm and can be reformulated as follows:

$$\mathcal{L}_\infty(\vec{x}-\vec{y}) = \max_{i=1}^{L} |x_i - y_i|$$

It is known that $\mathcal{L}_2$ norm is optimal (in the Maximum Likelihood sense) when measurement errors are additive, *i.i.d.* (independent, identically distributed) Gaussian [22]. It has been the most popular dissimilarity measure in similar time sequence matching [1, 9, 11, 19]. Linear correlation coefficient was used in [15], but it can be effectively converted to $\mathcal{L}_2$ norm without loss of information [8].

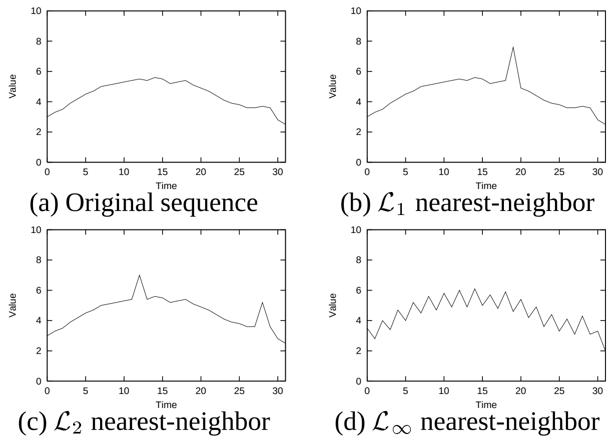


Figure 1: Different characteristics of $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_\infty$

$\mathcal{L}_1$ norm is optimal when measurement errors are additive, *i.i.d.* Laplacian (or Double Exponential), hence more robust against impulsive noise [22]. $\mathcal{L}_1$ has been used in the context of robust (parametric or non-parametric) regression [14, 20, 22] for many applications including time sequences [14, 6]. More recently, it was also used in [10] for their hashing-based similarity search technique.

$\mathcal{L}_\infty$ was used for atomic matching in a more complex dissimilarity measure in [2]. The measure proposed in [2], however, only decides whether two sequences are similar or not and ranking of query result is not possible.

Figure 1 illustrates the characteristics of different $\mathcal{L}_p$ norms. All sequences are of length 32. The original sequence is in (a). We added a single impulse of size 2.5 in (b), two impulses of size 1.5 in (c). In (d), we added and subtracted 0.5 alternately at each time spot. Then the closest sequences to the sequence (a) with respect to $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_\infty$, are (b), (c), and (d), respectively. This example clearly shows the different notion of similarity each norm offers.

Effective feature extraction functions such as DFT and DWT are available only for $\mathcal{L}_2$ because they are rotation-based (orthonormal transformations) and do not preserve distance for $\mathcal{L}_1$ and $\mathcal{L}_\infty$ in the feature space. Thus, in case of $\mathcal{L}_\infty$, they were forced to search in high dimensional space [2], rather than low dimensional feature space as in [1, 9].

We believe the choice of appropriate dissimilarity measures is highly application dependent and up to application engineers. Since, however, the perspectives of different users can vary even on the same dataset, some form of *multi-modality* is required. In such an environment, a DBMS for similarity-based retrieval of time sequences must provide a single unified framework which supports:

- multiple similarity models simultaneously,
- indexing for fast retrieval, and,
- re-use of the same index structure for multiple models.

In this regard, our first goal is to provide a general indexing scheme which can be used for any of $\mathcal{L}_p$ norms

($p = 1, 2, \ldots, \infty$). We specifically support the following two types of queries.

Problem 3.1 ($\mathcal{L}_p$-based Whole Sequence Matching) *Given a query sequence $\vec{q}$ and a set of sequences SEQ ($|\vec{q}| = |\vec{x}|$, for all $\vec{x} \in$ SEQ), find all sequences $\vec{x}$ in SEQ such that $\mathcal{L}_p(\vec{q} - \vec{x}) \leq \epsilon$, for any value of $p = 1, 2, \ldots, \infty$.*

Problem 3.2 ($\mathcal{L}_p$-based Subsequence Matching) *Given a query sequence $\vec{q}$ and a set of sequences SEQ ($|\vec{q}| \leq |\vec{x}|$, for all $\vec{x} \in$ SEQ), find all subsequences $\vec{x}'$ of all $\vec{x}$ in SEQ such that $|\vec{q}| = |\vec{x}'|$ and $\mathcal{L}_p(\vec{q} - \vec{x}') \leq \epsilon$, for any value of $p = 1, 2, \ldots, \infty$.*

While we are primarily concerned with *whole* and *subsequence* matching, we note that a fast method for both types of matching is also essential for more complex matching such as the one proposed in [2], in which *atomic* matching is in fact whole matching based on $\mathcal{L}_\infty$.

Some transformations can be allowed before sequences are compared. These include *offset translation*, *amplitude scaling* [11, 2, 15], and *time scaling* [19]. Offset translation subtracts/adds a certain offset value (usually mean) from each element of a sequence. Amplitude scaling multiplies a normalization factor to the element such that either the amplitude is within a fixed range or the sample variance is 1. Time scaling is to enlarge the time axis by a certain amount so that two sequences of different lengths can be matched. They provide a certain degree of flexibility in the notion of similarity. Our next goal is to support these transformations in our scheme.

Problem 3.3 (Transformations) *Support efficiently 'offset translation', 'amplitude scaling', and 'time scaling' in our indexing scheme.*

3.1 Proposed Method – Segmented Means

Suppose we have a set of sequences of length L. The basic idea of our proposal consists of two steps. First we partition each time sequence into s segments of equal length l. We assume $L = s * l$. Otherwise, we add zeros at the end of sequences. Note that it does not affect query results. Next, we extract simple features from each segment. We propose to use *mean* as a feature for all $\mathcal{L}_p$ norms.

Formally, let $\vec{x} = \langle x_1, \ldots, x_L \rangle$ be a sequence of length L. Let s and l be two numbers such that $L = s * l$. Then $\vec{x}$ can be divided into s segments of length l. Let P_j^x denote the j-th segment of $\vec{x}$, *i.e.*,

$$P_j^x = \langle x_{(j-1)l+1}, \ldots, x_{j \cdot l} \rangle.$$

We define a feature vector of $\vec{x}$ as follows. (See Figure 2 for an example.)

Definition 3.1 (Segmented-Mean Feature) *Given a sequence $\vec{x} = \langle x_1, \ldots, x_L \rangle$ and the number of segments $s > 0$, define the feature vector $\vec{F_s^x}$ of $\vec{x}$ by,*

$$\vec{F_s^x} = \langle f_1^x, \ldots, f_s^x \rangle = \langle mean(P_1^x), \ldots, mean(P_s^x) \rangle$$

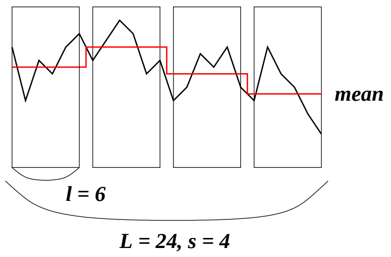


Figure 2: Example of Segmented Means

The algorithm to compute $\vec{F_s^x}$ is fairly obvious and omitted in this paper. To avoid the possibility of false dismissals, we must show that the distance between feature vectors lower-bounds that of original sequences. It is not very hard to see that it is indeed the case, *i.e.*, for all $p = 1, \ldots, \infty$,

$$\mathcal{L}_p(\vec{F_s^x}) \leq \mathcal{L}_p(\vec{x})$$

In practice, however, $\vec{F_s^x}$ is a poor approximation of $\vec{x}$, since it is essentially a down-sampling of $\vec{x}$. Much of the information would be lost and, consequently, too many false alrams would occur.

Our goal is to find a way to compensate the loss of information so that we could reduce the number of false alarms. More specifically, we seek a factor $\alpha_p > 1$ such that,

$$\alpha_p \cdot \mathcal{L}_p(\vec{F_s^x}) \leq \mathcal{L}_p(\vec{x})$$

We claim that there exists such a factor, thanks to the nice mathematical property of $\vec{F_s^x}$, and we will take advantage of it for efficient query processing. There is a well-known mathematical result on *convex* functions. We borrow the following theorem from [17, p.379].[1](See Figure 3 for an intuitive example.)

Theorem 3.1 *Suppose that $x_1, \ldots, x_L \in \mathcal{R}$, and $\lambda_1, \ldots, \lambda_L \in \mathcal{R}$ such that $\lambda_i \geq 0$ and $(\sum_{i=1}^L \lambda_i) = 1$. If f is a convex function on $\mathcal{R}$, then*

$$f(\lambda_1 x_1 + \cdots + \lambda_L x_L) \leq \lambda_1 f(x_1) + \cdots + \lambda_L f(x_L)$$

where $\mathcal{R}$ is the set of real numbers.

It is clear that $f(\cdot) = |\cdot|^p$ is a convex function on $\mathcal{R}$ for $1 \leq p < \infty$. Hence, as a direct consequence of Theorem 3.1 by taking $\lambda_i = \frac{1}{L}$, we have the following corollary.

Corollary 3.2 *For any sequence $\vec{x} = \langle x_1, \ldots, x_L \rangle$ and $1 \leq p < \infty$, the following holds.*

$$L \cdot |mean(\vec{x})|^p \leq \sum_{i=1}^{L} |x_i|^p$$

Or, equivalently, for each segment of $\vec{x}$, we have, for $1 \leq j \leq s$,

$$l \cdot |mean(P_j^x)|^p \leq \sum_{i=(j-1)l+1}^{j \cdot l} |x_i|^p$$

[1]The definitions of convex sets and functions are beyond the scope of the paper and are found in [17, pp.373-376]. Note also that we modified it such that we only consider $\mathcal{R}$ rather than $\mathcal{R}^d$.

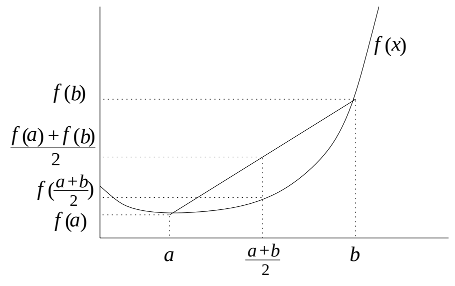


Figure 3: Illustration of convex function theorem

Now we have our main theorem as follows.

Theorem 3.3 *For any sequence $\vec{x} = \langle x_1, \ldots, x_L \rangle$ and $1 \le p \le \infty$, the following holds.*

$$\sqrt[p]{l} \cdot \mathcal{L}_p(\vec{F_s^x}) \le \mathcal{L}_p(\vec{x})$$

Proof: We first consider when $p \neq \infty$. By the definitions of $\mathcal{L}_p$ and $\vec{F_s^x}$,

$$\begin{aligned} l \cdot \mathcal{L}_p(\vec{F_s^x})^p &= l \cdot \sum_{j=1}^{s} |\text{mean}(P_j^x)|^p \\ \text{By Corollary 3.2,} \quad &\le \sum_{j=1}^{s} \left(\sum_{i=(j-1)l+1}^{j \cdot l} |x_i|^p \right) \\ &= \sum_{i=1}^{L} |x_i|^p \\ &= \mathcal{L}_p(\vec{x})^p \end{aligned}$$

By taking p-th root on both sides, we prove the theorem. If $p = \infty$, then

$$\begin{aligned} \mathcal{L}_\infty(\vec{F_s^x}) &= \max_{j=1}^{s} |\text{mean}(P_j^x)| \\ &\le \max_{j=1}^{s} \left(\max_{i=(j-1)l+1}^{j \cdot l} |x_i| \right) \\ &= \max_{i=1}^{L} |x_i| \;=\; \mathcal{L}_\infty(\vec{x}) \end{aligned}$$

Since $\sqrt[\infty]{l} = 1$, it completes the proof. □

3.1.1 Query Processing

Thanks to Theorem 3.3, we can efficiently handle ϵ-range queries with segmented-mean feature vectors. Suppose we are to compare two sequences $\vec{x}$ and $\vec{y}$. By the theorem, we know that $\mathcal{L}_p(\vec{x} - \vec{y}) \le \epsilon$ implies $\mathcal{L}_p(\vec{F_s^x} - \vec{F_s^y}) \le \epsilon / \sqrt[p]{l}$. (The converse does not hold in general.) Therefore, any $\mathcal{L}_p$-based ϵ-range queries against a set of sequences $\vec{x}$ can be correctly converted to $\mathcal{L}_p$-based $(\epsilon / \sqrt[p]{l})$-range queries against a set of the corresponding feature vectors $\vec{F_s^x}$ without worrying about the possibility of false dismissals. This is an improvement to the plain usage of the feature vectors, since we have reduced the search range by a factor of $\sqrt[p]{l}$.

We summarize a general strategy for the whole-sequence matching (Problem 3.1) as follows:

1. Extract feature vectors $\vec{F_s^x}$ for all $\vec{x} \in$ SEQ. The number of segments, s, is a system tuning parameter as the number of Fourier coefficients in [1]. (Trailing zeros are padded if necessary.)
2. Build an index structure on $\vec{F_s^x}$ using any of the readily available multi-dimensional access methods such as the R-tree.
3. For each $\mathcal{L}_p$-based ϵ-range query with a query sequence $\vec{q}$, extract $\vec{F_s^q}$ and convert the query to an equivalent $(\epsilon / \sqrt[p]{l})$-range query with $\vec{F_s^q}$ in the feature space, and perform search on the index. ($l = \lceil L/s \rceil$)
4. Filter out false alarms.

Processing subsequence matching queries (Problem 3.2) is more complex. The basic idea is fairly the same as in [9]. We assume that all sequences including query sequences are longer than a predetermined minimum length 'w'. (In this case, however, the length of each individual sequence can vary.) Then, our strategy is the following:

1. A sequence $\vec{x}$ is divided into $(|\vec{x}| - w + 1)$ sliding windows of fixed length w and extract the segmented-mean features from them. We use the same value of s as in the case of whole-sequence matching.
2. The feature vectors form a trail in the s-dimensional feature space. To reduce the storage overhead and enhance the system performance, we divide them into a few sub-trails based on the *'marginal cost'* criterion defined in [9], and compute their minimum bounding rectangles (MBRs).
3. We repeat the above steps for each $\vec{x} \in$ SEQ.
4. Build an index structure on the MBRs using any of the readily available multi-dimensional access methods such as the R-tree.
5. For each $\mathcal{L}_p$-based ϵ-range query with a query sequence $\vec{q}$,
   (a) Divide $\vec{q}$ into $p(= \lfloor |\vec{q}|/w \rfloor)$ non-overlapping subsequences, $\vec{q_k}$, $1 \le k \le p$. (Note that we can ignore the remaining $(|\vec{q}| - p \cdot w)$ elements without compromising the correctness of the query results.)
   (b) For each feature vector of $\vec{q_k}$, perform $(\epsilon / \sqrt[p]{l \cdot p})$-range search on the index and 'OR' the query results.
6. Filter out false alarms.

3.1.2 Transformations - Data Preprocessing

We have shown how to efficiently process $\mathcal{L}_p$-based queries using the proposed feature extraction method. We now turn to our next goal. As for offset translation and amplitude scaling, let $\vec{y} = a \cdot \vec{x} - b$ with 'a' for the scaling factor

```
Algorithm HaarWaveletCoefficients
  Input:  X[1,...,$L$], $L = 2^n$ for some $n$
  Output:  H[1,...,$L$], Haar wavelet coeff. for X[]

  H[1,...,$L$] := X[1,...,$L$];
  For (len := $L$; len >= 2; len := len/2) {
    For (i := j := 1; i < len; i := i+2, j := j+1) {
      S[j] := (H[i] + H[i+1]) / $\sqrt{2}$;
      D[j] := (H[i] - H[i+1]) / $\sqrt{2}$;
    }
    For (i := 1; i <= len/2; i := i+1) {
      H[i] := S[i];
      H[i+(len/2)] := D[i];
    }
  }
  Return H[1,...,$L$];
End Algorithm
```

Figure 4: Algorithm to compute Haar wavelet coefficients

and 'b' for the offset value. That is, $\vec{y}$ is the translated and rescaled version of $\vec{x}$. Then the feature vector of $\vec{z}$ can be computed as follows:

$$\begin{aligned} \vec{F_s^y} &= \langle \text{mean}(P_1^y), \ldots, \text{mean}(P_s^y) \rangle \\ &= \langle (a \cdot \text{mean}(P_1^x) - b), \ldots (a \cdot \text{mean}(P_s^x) - b) \rangle \\ &= a \cdot \vec{F_s^x} - b \end{aligned}$$

Thus, once we have the feature vectors, it is almost straightforward to compute the translated and rescaled versions.

As for time scaling, since extending time axis does not change the mean, the mean feature vector is *invariant* under time scaling. Let $\vec{z}$ is an extended version of $\vec{x}$ by 'c' times for an integer c. That is, $z_i = x_{\lceil i/c \rceil}$. Then,

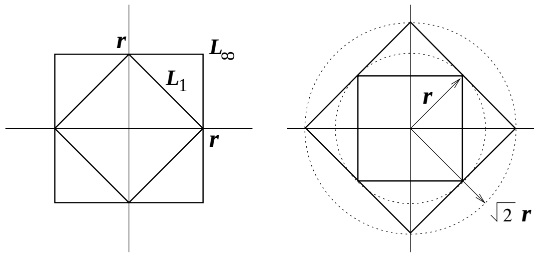
$$\begin{aligned} \vec{F_s^z} &= \langle \text{mean}(P_1^z), \ldots, \text{mean}(P_s^z) \rangle \\ &= \left\langle \frac{c \cdot \text{mean}(P_1^x)}{c}, \ldots, \frac{c \cdot \text{mean}(P_s^x)}{c} \right\rangle \\ &= \vec{F_s^x} \end{aligned}$$

Thus, we can reuse the same feature vector *as-is* for the time-extended version. We believe other transformations such as *moving-average* [19] can be easily handled likewise.

3.2 An Alternative–How to use DWT

As an alternative, we can use the existing feature extraction methods based on orthogonal linear transforms such as DWT and DFT. In this paper, we focus on DWT, especially '*Haar*' DWT, since it has been widely used as a state-of-the-art method for various database applications recently [16, 23]. We present an algorithm to compute 1-d Haar wavelet coefficients in Figure 4. (For the theory of wavelets, readers are referred to [5].)

Since DWT as well as DFT is an orthogonal linear transform, it rotates the data distribution in a predetermined way. As such, $\mathcal{L}_2$ norm is perfectly preserved by the transform, since it is invariant under rotations. Other $\mathcal{L}_p$ norms for



(a) Before rotation (b) After rotation

Figure 5: Example of adjusting ranges for rotated points

$p \neq 2$ are not preserved. Therefore, we can not use the DWT-based feature vectors for arbitrary $\mathcal{L}_p$-based query processing.

One way to fix the problem is to adjust the search range such that all qualifying sequences are included within the search boundary. Suppose $\vec{x}'$ and $\vec{y}'$ are the rotated versions of two sequences $\vec{x}$ and $\vec{y}$ of length L, respectively. Then, it is not hard to see the following translation rules hold: (Figure 5 describes the idea in the 2-d plane, *i.e.*, $L = 2$.)

$$\begin{aligned} \mathcal{L}_1(\vec{x} - \vec{y}) \leq \epsilon &\implies \mathcal{L}_2(\vec{x}' - \vec{y}') \leq \epsilon \\ \mathcal{L}_\infty(\vec{x} - \vec{y}) \leq \epsilon &\implies \mathcal{L}_2(\vec{x}' - \vec{y}') \leq \sqrt{L} \cdot \epsilon \end{aligned}$$

Similar rules are possible for $\mathcal{L}_p$ norms for $3 \leq p < \infty$.

We convert each $\mathcal{L}_1$- and $\mathcal{L}_\infty$-based ϵ-range queries to $\mathcal{L}_2$-based queries with search ranges according the above conversion rules, and then perform search on the index built on top of the DWT feature vectors. Note that, this way, we can guarantee no false dismissals. As we will see in the next section, however, it is not very efficient except for $\mathcal{L}_2$ norm.

3.2.1 The Haar DWT vs the Segmented Means

The two types of feature vectors produced by the proposed method and the Haar DWT are closely related to each other. More specifically, we have the following theorem.

Theorem 3.4 *Let $\vec{H_s^x}$ denote a feature vector of $\vec{x}$, composed of the first s Haar wavelet coefficients. We further assume $|\vec{x}| = 2^n$ ($n > 0$) and $s = 2^m$ ($n \geq m \geq 0$). Then, the following equality holds.*

$$\mathcal{L}_2(\sqrt{l} \cdot \vec{F_s^x}) = \mathcal{L}_2(\vec{H_s^x})$$

where $l = |\vec{x}|/s = 2^{n-m}$.

Proof: By definition,

$$\begin{aligned} \sqrt{l} \cdot \vec{F_s^x} &= \langle \sqrt{l} \cdot \text{mean}(P_1^x), \ldots, \sqrt{l} \cdot \text{mean}(P_s^x) \rangle \\ &= \langle \text{sum}(P_1^x)/\sqrt{l}, \ldots, \text{sum}(P_s^x)/\sqrt{l} \rangle \end{aligned}$$

Note that, if we take the Haar DWT of the above vector, we get $\vec{H_s^x}$. Since the Haar DWT is invariant under $\mathcal{L}_2$, we have,

$$\begin{aligned} \mathcal{L}_2(\sqrt{l} \cdot \vec{F_s^x}) &= \mathcal{L}_2(\text{DWT}(\sqrt{l} \cdot \vec{F_s^x})) \\ &= \mathcal{L}_2(\vec{H_s^x}) \end{aligned}$$

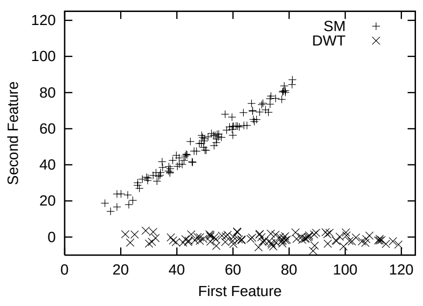


Figure 6: Examples of $\mathcal{S}_{SM}$ and $\mathcal{S}_{DWT}$

Hence, the theorem holds. □

Let $\mathcal{S}_{\text{SM}}$ and $\mathcal{S}_{\text{DWT}}$ be two feature spaces defined by $\sqrt{l} \cdot \vec{F}^x_s$ and $\vec{H}^x_s$, respectively. The above theorem tells us that if the dimension of feature space is some power of 2 (*i.e.*, $s = 2^m$), then the $\mathcal{S}_{\text{SM}}$ is a rotated version of $\mathcal{S}_{\text{DWT}}$ and *vice versa*. In Figure 6, we present examples of $\mathcal{S}_{\text{SM}}$ and $\mathcal{S}_{\text{DWT}}$ of 100 time sequences ($s = 2$). We observe that they are indeed rotated versions of each other. In terms of indexing, however, $\mathcal{S}_{\text{DWT}}$ seems a little bit better since its minimum bounding rectangle (MBR) is smaller as we can see in the example. Therefore, for $\mathcal{L}_2$-based queries, we expect slightly better performance from $\mathcal{S}_{\text{DWT}}$ and, as we will see later, the experimental results prove this point.

4 Experimental Results

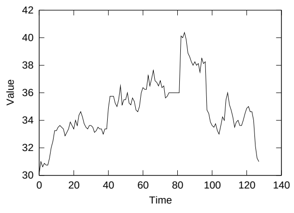
To verify the effectiveness of the proposed method, we performed experiments on real time sequences (daily stock prices) and synthetically generated time sequences. Our experiments were based on ϵ-range queries for both whole- and subsequence matching. We compared the proposed method and the DWT-based method as well as the sequential scanning method as a sanity checker. All methods were implemented in the C programming language. For R-tree, we used DR-tree (v2.5) library developed at the Univ. of Maryland with some modifications to handle $\mathcal{L}_p$-based search. As a measure of success, we recorded the wall clock time with the UNIX '`time`' command. All experiments were performed on a dedicated Sun UltraSparc-1 workstation with a 143MHz CPU, 64MB of memory and SCSI disks (Seagate ST410800N), running SunOS version 5.5 operating system.

We present more specific information on the experimental setting in the following subsection.

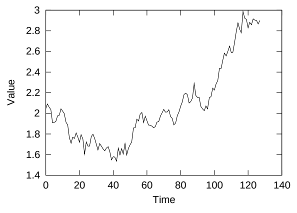
4.1 Experimental Setting

For the experiment, we prepared two datasets of sequences. Samples of these time sequences are plotted in Figure 7.

- STOCK: The stock dataset contains 675 stocks, each with varying number of daily closing prices. The average length is 1,187. For the whole-sequence matching, we generated fixed length sequences with 128 samples each. The sample windows overlap by $1/3$ of the window size. For subsequence matching, we used the dataset as-is.



(a) Stock



(b) Synthetic

Figure 7: Sample Time Sequences

Dataset	Num. of Seq.	(Avg.) Seq. Len.	Feature Dim.
STOCK	675	1,187	4
SYNTH	30,000	128	4

Table 3: Summary of the Experimental Setting

- SYNTH: Additional 30,000 synthetic sequences, with 128 samples each, were generated using the *random-walk* model following [1]. More specifically, each sequence was generated by the following formula.

$$x_t = x_{t-1} + \alpha \cdot z_t$$

where $x_0 \sim \text{Uniform}(2, 10)$, $z_t \sim \text{Normal}(0, 1)$, and $\alpha = 0.06$, for $t = 1, \ldots, 128$.

We compared the following 3 methods:

- SM: Our proposed method based on the segmented means.
- DWT: The alternative method based on the Haar wavelet transform.
- SCAN: The naive sequential scanning method.

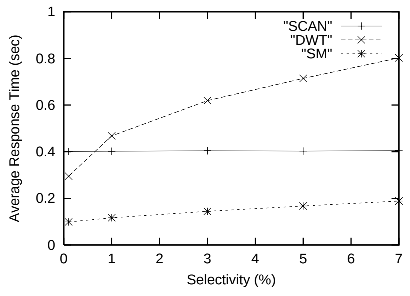
An important parameter is the dimension of the feature space, *i.e.*, the length of feature vectors. In general, the optimal value depends on the datasets, the feature extraction methods to use, and the distance functions ($\mathcal{L}_p$ norms). We fixed it as 4 because of the following two reasons:

- The value was good enough for both SM and DWT methods, a little more in favor of DWT, regardless of the datasets.
- Since one of our goals is to re-use the same index structure for all $\mathcal{L}_p$ norms, we need to fix it for all values of p.

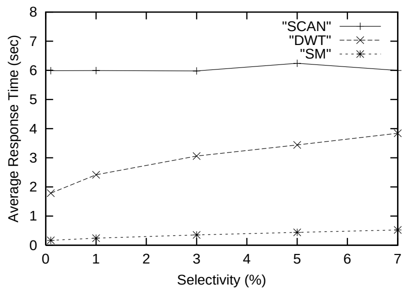
Table 3 summarizes the experimental setting.

4.2 Whole Sequence Matching Queries

We first performed $\mathcal{L}_p$-based whole-sequence matching queries. We took 100 sequences randomly from each dataset and used them as the query sequences. We measured the average response time (including both the search time and the post-processing time). Search ranges were chosen such that the average selectivity of query results be

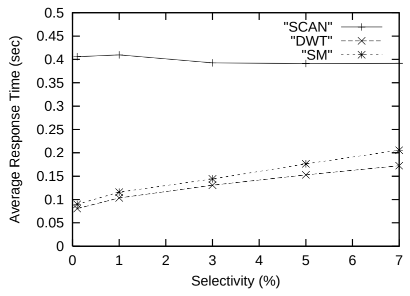


(a) Stock

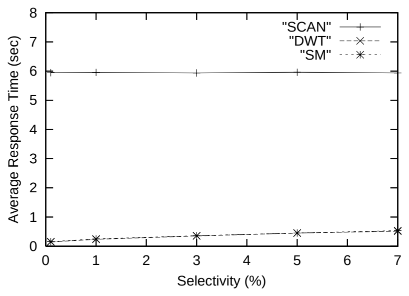


(b) Synthetic

Figure 8: $\mathcal{L}_1$-based whole-sequence matching



(a) Stock



(b) Synthetic

Figure 9: $\mathcal{L}_2$-based whole-sequence matching

0.1%, 1%, 3%, and 7%, for each value of $p = 1, 2, \infty$ and for each dataset.

The results of $\mathcal{L}_1$-based whole matching queries are presented in Figure 8. The proposed method is the clear winner for both datasets. Interestingly, DWT method was even slower than the SCAN method on the stock dataset. For the synthetic dataset, the proposed method achieved 10 time speedup over the DWT method and 50 time speedup over the naive method at the selectivity of 0.1%.

In the case of $\mathcal{L}_2$-based queries (see Figure 9), the DWT method was slightly faster than the proposed method for the stock dataset. It is, however, expected because DWT is highly optimized for $\mathcal{L}_2$ norm. For the synthetic dataset, it is almost impossible to distinguish between the proposed method and the DWT method and they both scaled up very well.

Figure 10 presents the results from $\mathcal{L}_\infty$-based queries. Again, the proposed method is the winner for both datasets, although the difference between the proposed method and the DWT method is small in the stock dataset. For the synthetic dataset, they both scaled very well and the proposed method consistently outperformed the competitor.

As a summary, we conclude that the proposed method is the clear winner in all cases except for $\mathcal{L}_2$-based queries against the stock dataset. But the DWT method performed very poorly for $\mathcal{L}_1$-based queries.

4.3 Subsequence Matching Queries

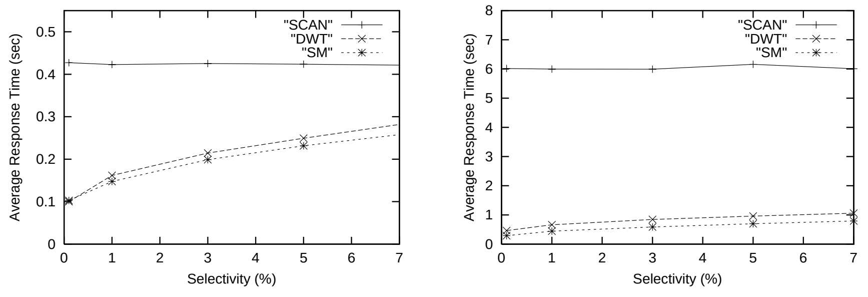
We next performed $\mathcal{L}_p$-based subsequence matching queries. For the stock dataset, we re-used the same query sequences that had been used for the whole matching queries. For the synthetic dataset, we used the first 64 values of the query sequences from the whole matching experiment. Hence the ratios between the query length and the data length are 128:1187 for the stock dataset and 64:128 for the synthetic dataset. The search ranges were chosen in the same way as in the previous experiment.

In Figure 11, presented are the results from $\mathcal{L}_1$-based subsequence queries. We observed that, again, the proposed method is the clear winner for both datasets and it scales very well even at the relatively hight selectivity (7%). The DWT performed poorly for both datasets. At highest selectivity, it almost converged to the naive method.

In Figure 12, $\mathcal{L}_2$-based query results are shown. Yet gain, the DWT method performed slightly better than the proposed method on the stock dataset, but the difference was small. For the synthetic dataset, the proposed method outperformed the DWT method just a little after 1% of selectivity. On both datasets, they scaled very well.

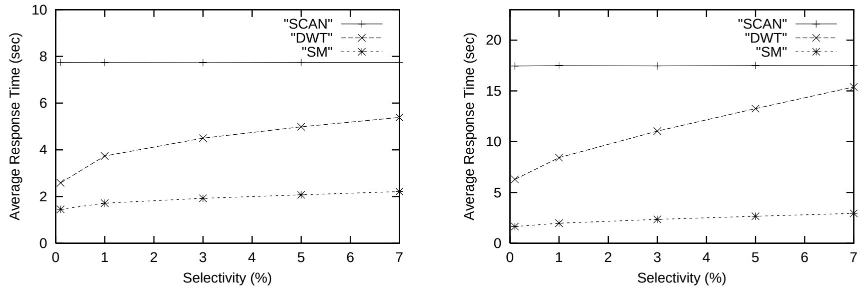
In the case of $\mathcal{L}_\infty$ (Figure 13), the results are not much different from the other cases and the proposed method consistently outperformed the competitor.

Summary In Table 4, we present the relative response time of the proposed method *w.r.t.* the DWT-based method for the 3%-selectivity queries. For $\mathcal{L}_1$-based queries, the proposed method outperforms the DWT-based method by big margins, in all datasets. For $\mathcal{L}_2$-based queries, the DWT-based is slightly faster as we anticipated. Note, however, the proposed method outperforms in subsequence queries against the synthetic dataset. For $\mathcal{L}_\infty$-based queries, the proposed method consistently outperforms by the competitor by up to 30% margin. Overall, we conclude



(a) Stock (b) Synthetic

Figure 10: $\mathcal{L}_\infty$-based whole-sequence matching



(a) Stock (b) Synthetic

Figure 11: $\mathcal{L}_1$-based subsequence matching

Dataset	Whole			Subsequence		
	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_\infty$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_\infty$
STOCK	0.23	1.1	0.93	0.43	1.05	0.88
SYNTH	0.12	1.03	0.7	0.18	0.95	0.74

Table 4: Relative response time (T_{SM}/T_{DWT}) for 3%-selectivity queries

that the proposed method is the winner in the competition.

5 Conclusion

The major contribution of the paper is two-folds:

- **Multi-modality Support:** No single model of similarity is suitable for every application. Sometimes several similarity models may be required for the same database of sequences, depending on the different perspectives of different users. We addressed this problem by supporting arbitrary $\mathcal{L}_p$ norms for any value of $p = 1, 2, \ldots, \infty$, because they are the most popular class of dissimilarity measures and, also, they can be used as the building blocks for more complex ones. No previous work has proposed a single framework to support this *multi-modal* query processing for time sequences.

- **Efficient Indexing:** For efficient query processing, we proposed a new unified indexing scheme which provides the following advantages over previous approaches.
  - All $\mathcal{L}_p$ norms are supported simultaneously.
  - Indexing for fast retrieval is supported.
  - The same index structure can be re-used for different $\mathcal{L}_p$ norms.
  - It is easy to incorporate such data preprocessing techniques as ‘offset translation’, ‘amplitude scaling’, and ‘time scaling’.

We showed the soundness of our method mathematically. We also explained in detail how to efficiently process both the whole-sequence and the subsequence matching queries in our unified indexing scheme. Through extensive experiments, we verified that our method is very efficient. Our method achieved up to 10 time speedup over the DWT-based state-of-the-art method and scaled up very well for all $\mathcal{L}_p$ norms on both the real and the synthetic datasets. Further research will focus on extending the proposed method to a broader class of similarity models.

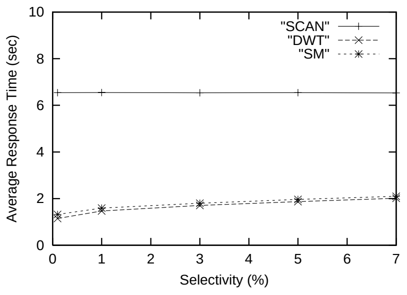
References

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proc. of the FODO Conf.*, Evansotn, IL, October 1993.

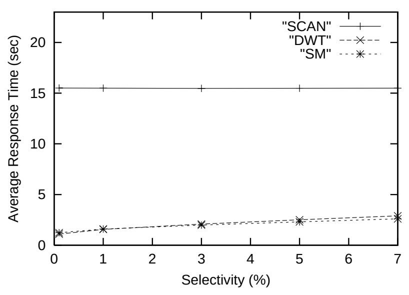
[2] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series database. In *Proc. of the VLDB Conf.*, Zürich, Switzerland, 1995.

[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 *Proc. ACM SIGMOD Conf.*, pages 322–331, Atlantic City, NJ, May 1990.

[4] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. In *Proc. VLDB Conf.*, Bombay, India, 1996.

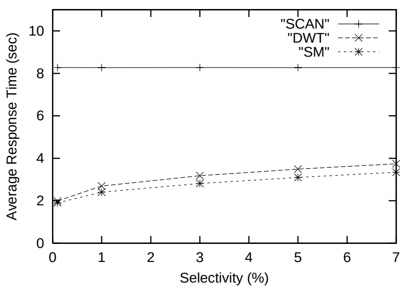


(a) Stock

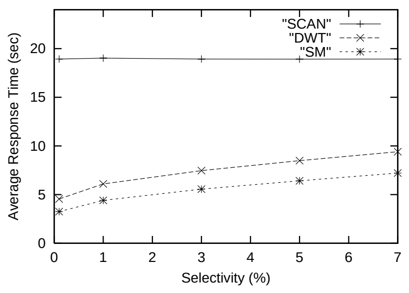


(b) Synthetic

Figure 12: $\mathcal{L}_2$-based subsequence matching



(a) Stock



(b) Synthetic

Figure 13: $\mathcal{L}_\infty$-based subsequence matching

[5] I. Daubechies. *Ten Lectures on Wavelets*. Capital City Press, Montpelier, Vermont, 1992. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[6] Y. Dodge, editor. *L_1-Statistical Procedures and Related Topics*. Institute for Mathmatical Statistics, Hayward, CA, 1997.

[7] C. Faloutsos, H. V. Jagadish, A. O. Mendelzon, and T. Milo. A signature technique for similarity-based queries. In *Proc. of SEQUENCES'97*, Salerno, Italy, June 1997.

[8] C. Faloutsos and K.-Ii. Lin. FastMap: a Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. *Proc. of ACM-SIGMOD*, pages 163–174, May 1995.

[9] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast Subsequence Matching in Time-Series Databases. In *Proc. of the ACM SIGMOD Conf.*, May 1994.

[10] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proc. of VLDB Conf.*, Edinburgh, Scotland, 1999.

[11] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proc. of Constraint Programming 95*, Marseilles, France, September 1995.

[12] A. Guttman. R-Trees: a Dynamic Index Structure for Spatial Searching. In *Proc. ACM SIGMOD Conf,*, pages 47–57, Boston, Mass, Jun 1984.

[13] E. J. Keogh and M. J. Pazzani. A simple dimensionality reduction technique for fast similairity search in large time series databases. In *Proc. of the 4th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, Kyoto, Japan, 2000.

[14] K. D. Lawrence and J. L. Arthur, editors. *Robust Regression*. Dekker, 1990.

[15] C.-S. Li, P. S. Yu, and V. Castelli. HierarchyScan: A Hierarchical Similarity Search Algorithm for Databases of Long Sequences. In *Proc. of ICDE*, pages 546–553, New Orleans, LA, 1996.

[16] A. Natsev, R. Rastogi, and K. Shim. WARLUS: A Similarity Retrieval Algorithms for Image Databases. In *Proc. of ACM SIGMOD Conf.*, pages 395–406, Philadelphia, PA, 1999.

[17] M.H. Protter and C.B. Morrey. *A First Course in Real Analysis*. Springer-Verlag, 1977.

[18] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[19] D. Rafiei and A. Mendelzon. Similarity -based queries for time series data. In *Proc. of the ACM SIGMOD Conf.*, Tucson, AZ, May 1997.

[20] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection*. John Wiley, New York, 1987.

[21] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+ Tree: a Dynamic Index for Multi-Dimensional Objects. In *Proc. VLDB Conf.*, pages 507–518, 1987.

[22] N. D. Sidiropoulos and R. Bros. Mathematical Programming Algorithms for Regression-based Non-linear Filtering in R^N. *IEEE Trans. on Signal Processing*, Mar 1999.

[23] J. S. Vitter and M. Wang. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *Proc. of ACM SIGMOD Conf.*, pages 193–204, Philadelphia, PA, 1999.

[24] B.-K. Yi, H.V. Jagadish, and C. Faloutsos. Efficient Retrieval of Similar Time Sequences under Time Warping. In *IEEE Proc. of ICDE*, Feb 1998.