Data structures and algorithms – Haar wavelets for efficient similarity search of time series: With and without time warping

ARTICLE *in* IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING · JUNE 2003

Impact Factor: 2.07 · DOI: 10.1109/TKDE.2003.1198399 · Source: IEEE Xplore

Haar Wavelets for Efficient Similarity Search of Time-Series: With and Without Time Warping

Franky Kin-Pong Chan, Ada Wai-chee Fu, *Member*, *IEEE*, and Clement Yu

Abstract—We address the handling of time series search based on two important distance definitions: Euclidean distance and time warping distance. Conventional method reduces the dimensionality by means of Discrete Fourier Transform. We apply the Haar Wavelet Transform technique and propose the use of a proper normalization so that the method can guarantee no false dismissal for Euclidean distance. We found that this method has competitive performance from our experiments. Euclidean distance measurement cannot handle the time shifts of patterns. It fails to match the same rise and fall patterns of sequences with different scales. A distance measure that handles this problem is the time warping distance. However, the complexity of computing the time warping distance function is high. Also, as time warping distance is not a metric, most indexing techniques would not guarantee any false dismissal. We propose efficient strategies to mitigate the problems of time warping. We suggest a Haar wavelet-based approximation function for time warping distance, called Low Resolution Time Warping, which results in less computation by trading off a small amount of accuracy. We apply our approximation function to similarity search in time series databases, and show by experiment that it is highly effective in suppressing the number of false alarms in similarity search.

Index Terms—Similarity search, time warping, wavelets, dimension reduction, multidimensional index, time series database, data mining.

1 Introduction

Time series data are of growing importance in many new database applications, such as data warehousing and data mining [3], [6], [2]. A time series (or time sequence)[1] is a sequence of real numbers, each number representing a value for an interested attribute at a time point. Typical examples include stock prices or currency exchange rates, the volume of product sales, biomedical measurements, weather data, etc., collected over time. Hence, time series database systems supporting fast retrieval of time series data and similarity queries are desired. In particular, for time series in finance, time series retrieval can help to identify "pair trading," the goal is to identify pairs (or groups) of stocks whose prices track one another (their difference is a stationary process), it can also help to find correlation of stocks over certain time interval, and to find patterns in temporal data for data mining. These issues about financial time series are discussed in [28].

In order to depict the similarity between two time series, we have to define a similarity measurement during the matching process. Given two time series $\vec{x} = \{x_0, x_1, \ldots, x_{n-1}\}$ and $\vec{y} = \{y_0, y_1, \ldots, y_{n-1}\}$, a standard approach is to compute the Euclidean distance $D(\vec{x}, \vec{y})$ between time series $\vec{x}$ and

$$\vec{y} : D(\vec{x}, \vec{y}) = \left(\sum_{i=0}^{n-1} |x_i - y_i|^2 \right)^{\frac{1}{2}}.$$

We can retrieve similar time series by considering distance $D(\vec{x}, \vec{y})$. In order to support efficient retrieval and matching of time series, we resort to indexing.

For index creation, the time series are preprocessed, the preprocessing may include normalization, transformation, noise reduction, etc., to produce a set of feature vectors. The feature vectors may also undergo a dimensionality reduction in order to achieve reasonable performance in the query evaluation. The resulting vectors are then inserted into a multidimensional index tree, on which users can pose queries. Upon the arrival of a query, preprocessing is done the same way as for the time series database in index creation. The feature vector acquired is used as a key to search the index tree, which results in a set of candidate sequences. Nonqualified sequences or *false alarms* are filtered in a postprocessing step by matching with the query sequence in time domain using full dimensionality. Qualified sequences are thus reported to the user.

In the above scenario, the time series typically undergo a dimensionality reduction, and most previous work has suggested to use Discrete Fourier Transform (DFT). DFT has been used in many other disciplines such as image processing, computer graphics, and signal processing. In recent years, it has been shown that another method based on discrete wavelet transform can be more effective than DFT. In the first part of this paper, we propose a wavelet-based

1. We shall use the terms *time series* and *time sequence* interchangeably.

- *F.K.-P Chan and A.W.-c. Fu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong. E-mail: fkpchan@alumni.cuhk.net and adafu@cse.cuhk.edu.hk.*
- *C. Yu is with the Department of Electrical Engineering and Computer Science, The University of Illinois at Chicago, Chicago, IL 60612. E-mail: yu@eecs.uic.edu.*

Manuscript received 8 June 2000; revised 17 Mar. 2001; accepted 26 July 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 112248.

method for dimension reduction for time-series similarity search. We use the Haar wavelets and show that with some normalization, the transformation can preserve Euclidean distance, which ensures that the similarity search would not give rise to false dismissal. Both range querying and nearest-neighbor querying are considered. We show that this method can lead to good performance by experiments on real stock data and synthetic data.

Other similarity models may also be employed. The choice of an appropriate model, in accordance with a particular kind of time series, gives rise to better interpretation and semantics of the similarity between two sequences. In the speech recognition field, *time warping* techniques [23], [6] are used extensively for similarity matching. The time warping distance allows fluctuating time scaling. We find that this can be useful for stock data. The question of similarity search with time warping is mentioned in [28].

Algorithms for time warping are proposed in [27], [26], [19]. Although they are able to match sequences with time shifts, the computations involved are rather costly. This imposes a problem in using this technique in an online system where prompt response is demanded. In principle, an index tree supporting time warping distance could be built for time series search. Unfortunately, unlike Euclidean distance, time warping distance does not satisfy the triangle inequality, which is a basic assumption in most indexing methods. The consequence is that we are unable to guarantee the retrieval of all qualified sequences for a given query, in other words, it may give rise to *false dismissals*. Some time warping based index techniques are described in Section 2.3 and their drawbacks are identified. We propose a novel approximation function based on wavelets for time warping distance, which results in lower time complexity by trading off a tiny amount of accuracy. This approximation function can be employed as the filtering function in the postprocessing step of the querying, which will be shown in Section 6 to be both effective and efficient experimentally.

The rest of this paper is organized as follows: In Section 2, we give some background information, including the wavelet transform. In Section 3, we present the idea of wavelet transform for the Euclidean distance based similarity measure. We then describe how to compute the time warping distance function and a method based on K-L transform with a lower bound filter in Section 4. In Section 5, we present our strategy in approximating time warping distance between two time series. In Section 6, performance evaluations together with scalability test are given. We give a conclusion Section 7.

2 Related Work

Time series is primarily concerned with the study of the time variations of a process. The states for a duration of n time units of a process can be represented by a vector of real numbers

$$x = [x_0\ x_1\ \cdots x_t\ \cdots x_{n-1}]^T,$$

where x_t is the state recorded at time t. For the sake of convenience, we depict time series in the above as as $\vec{x} = \{x_0, x_1, \ldots, x_{n-1}\}$ unless otherwise specified.

2.1 Wavelet Transform

Wavelet Transform (WT), or *Discrete Wavelet Transform* (DWT) [7], [14], [10] has been found to be effective in replacing DFT in many applications in computer graphics [29], image [20], speech [1], and signal processing [5], [4]. We propose to apply this technique to dimensionality reduction for time series. DWT is a discrete version of WT for numerical signal. The potential application of DWT in this problem was pointed out in [17]. It is of value to conduct studies and evaluations on time series retrieval and matching by means of wavelets.

The advantage of using DWT is its *multiresolution* representation of signals. It has the *time-frequency localization* property. For instance, the wavelet representation of a musical score can tell when the tones occur and what their frequencies are. While DFT extracts the lower harmonics which represent the general shape of a time sequence, DWT typically encodes a coarser resolution of the original time sequence with its preceding coefficients.

We shall make use of a very basic wavelet transform called the Haar wavelets. Haar transform can be seen as a series of averaging and differencing operations on a discrete time function. We compute the average and difference between every two adjacent values of $f(t)$. The procedure to find the Haar transform of a discrete function $f(t) = \{9, 7, 3, 5\}$ is shown below.

Resolution	Averages	Coefficients
4	$\{9, 7, 3, 5\}$	
2	$\{8, 4\}$	$\{1, -1\}$
1	$\{6\}$	$\{2\}$

Resolution 4 is the full resolution of the discrete function $f(t)$. In Resolution 2, $\{8, 4\}$ are obtained by taking the average of $\{9, 7\}$ and $\{3, 5\}$ at Resolution 4, respectively. $\{1, -1\}$ are the differences of $\{9, 7\}$ and $\{3, 5\}$ divided by two, respectively. This process is continued until a resolution of 1 is reached. The Haar transform $H(f(t)) = \{c_0, d_{0,0}, d_{1,0}, d_{1,1}\} = \{6, 2, 1, -1\}$ is obtained which is composed of the last average value 6 and the coefficients found on the rightmost column, 2, 1, and -1. It should be pointed out that c_0 is the *overall average value* of the whole time sequence, which is equal to $(9 + 7 + 3 + 5)/4 = 6$. Different resolutions can be obtained by adding differences back to or subtract differences from averages. For instance, $\{8, 4\} = \{6 + 2, 6 - 2\}$, where 6 and 2 are the first and second coefficient, respectively. This process can be done recursively until the full resolution is reached.

For a time sequence of length 4, the Haar transformation, or *decomposition* can be found by (1). asfor DFT, the length of the signal is restricted to numbers which are powers of 2. Consider the example input signal $\vec{x} = \{x_0, x_1, x_2, x_3\}$ as a column vector with length $n = 4$, an intermediate transform vector $\vec{w}$ is obtained from $\vec{x}$ via the Haar transform matrix $\mathbf{H}$. The final transformed vector will be $\{c_0, d_{0,0}, d_{1,1}\}$

$$\begin{bmatrix} x'_0 \\ d_{1,0} \\ x'_1 \\ d_{1,1} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}. \quad (1)$$

The factor $1/2$ associated with the Haar transform matrix can be varied according to different *normalization*[2] conditions. After the first multiplication of $\vec{x}$ and H, half of the Haar transform coefficients can be found which are $d_{1,0}$ and $d_{1,1}$ in $\vec{w}$ interleaving with some intermediate coefficients x'_0 and x'_1. Actually, $d_{1,0}$ and $d_{1,1}$ are the last two coefficients of the Haar transform. x'_0 and x'_1 are then extracted from $\vec{w}$ and put into a new column vector $\vec{x'} = [\, x'_0 \quad x'_1 \quad 00\,]^T$. $\vec{x'}$ is treated as the new input vector for transformation. This process is done recursively until one element is left in $\vec{x'}$. In the above case with four elements, c_0 and $d_{0,0}$, which are the first two coefficients of the Haar transform, can be found in the second iteration.

2.2 Similarity Search under Euclidean Distance

To provide efficient querying, dimensionality of time series is typically reduced and a common technique is by means of Discrete Fourier Transform (DFT) [2], [11]. An index built by means of DFT is also called an **F-index** [2]. Suppose the DFT of a time sequence $\vec{x}$ is denoted by $\vec{X}$. There is a mapping of signals from n-dimensional time domain to n-dimensional frequency domain. For many applications such as stock data, the *low frequency components* are located at the first f_c coefficients of $\vec{X}$, for a small f_c, which represent the general trend of the time sequence $\vec{x}$. These coefficients together can be seen as a high-dimensional vector and can be indexed in a multidimensional index structure such as an R-Tree or R*-Tree for fast retrieval of time sequences. In most previous works, range querying is considered. A **range query** (or epsilon query) evaluation returns sequences with Euclidean distance within ϵ from the query point. In such a query, a query time sequence of length n and a tolerance ϵ are given. To resolve the query, n-point DFT is applied to the query sequence and the first f_c features are used for similarity matching by an F-index, which returns all sequences within Euclidean distance ϵ.

Parseval's Theorem [21] shows that the Euclidean distance between two signals $\vec{x}$ and $\vec{y}$ in the time domain is the same as the Euclidean distance in frequency domain. Therefore, F-index may raise false alarms, but guarantees no false dismissal. After a range query in the F-index, qualified sequences are then checked against the query sequence in the original time domain. This postprocessing step eliminates the false alarms.

Another method employed for dimension reduction is *Karhunen-Loeve* (K-L) transform [30]. (This method is also known as Singular Value Decomposition (SVD) [17], and is called Principle Component analysis in statistical literature.) Given a collection of n-dimensional points, we project them onto a k-dimensional subspace where $k < n$, maximizing the variances in the chosen dimensions. The key weakness of K-L transform is the deterioration of performance upon incremental update of the index, as the projection axes are predetermined (static) by the covariance matrix in the initial collection of feature vectors. Although the projection is optimal for a fixed set of vectors, after incremental updates, the projection matrix should be recalculated and the index tree has to be reorganized periodically to keep up the search performance. Efficient methods for incremental update of SVD-based index are discussed in [15].

2. The normalization factor can be $1/2$ or $1/\sqrt{2}$, depending on the domain of application.

In [18], a hierarchical similarity search technique is proposed. The *correlation coefficients* are used as the similarity measure between two sequences. This similarity eliminates the need to generate all the subsequences for matching using sliding windows as in [11]. The correlation coefficients can be efficiently computed by employing the convolution theorem of DFT. Moreover, correlation coefficient has a 1:1 correspondence with the Euclidean distance between normalized versions of the matching sequences. The authors also attempts to use DWT for hierarchical search. However, the low resolution comparison can give rise to false dismissals. The method is also unable to handle sequence matching with time scaling that are allowed by time warping consideration.

A similarity index supporting L_P norm is suggested in [31]. Segmented means are used for transformation which reduces the storage for features. Rules are also developed for translating L_1 and L_∞ into L_2 queries, that can be searched in an R-tree like index. Apart from segmented means, DWT has also been employed for feature extractions. The Haar wavelets transformation is applied and it is shown by experiment that the DWT method is efficient for L_2 while slow for L_1. However, no further investigation on the multiresolution properties of DWT has been made. Such properties can be very useful in approximating time series with the time-warping consideration, which we emphasize in the second part of this paper.

2.3 Similarity Search under Time Warping

The ability of time warping measurement to match sequences with time shifts makes it an important similarity model in speech recognition, since human speech consists of varying durations and paces. The time warping distance definition tries to handle this problem. This paradigm is also true for time series matching in financial sector. Stock sequences may be considered similar if they have the same rise (mountains) and fall (valleys) patterns, neglecting the different scales in the time axis. The time warping technique aligns the mountains and valleys as much as possible by expanding and compressing the time axis accordingly. It follows that time warping distance is a good measurement for similarity matching of stock sequences.

The time warping distance between two sequences $\vec{x}$ and $\vec{y}$ is defined as

$$\begin{aligned} &D_{timewarp}(\langle\rangle, \langle\rangle) = 0, \\ &D_{timewarp}(\vec{x}, \langle\rangle) = D_{timewarp}(\langle\rangle, \vec{y}) = \infty, \\ &D_{timewarp}(\vec{x}, \vec{y}) = D_{base}(Head(\vec{x}), Head(\vec{y})) \\ &+ \min \\ &\left\{ \begin{array}{ll} D_{timewarp}(\vec{x}, Rest(\vec{y})) & \text{x} - \text{stutter} \\ D_{timewarp}(Rest(\vec{x}), \vec{y}) & \text{y} - \text{stutter} \\ D_{timewarp}(Rest(\vec{x}), Rest(\vec{y})) & \text{no stutter} \end{array} \right\}, \end{aligned} \tag{2}$$

where $\langle\rangle$ denotes a null sequence, $Head(\vec{x})$ denotes the first element of $\vec{x}$, and $Rest(\vec{x})$ stands for the remaining elements of $\vec{x}$. D_{base} can be any of the distance functions, like the city-block distance, although our primary concern will be the Euclidean distance. Note that this definition does not require two sequences to be of the same length. Additional constraints [27] may also be applied to restrict the degree of freedom of the warping process for different applications.

Different Searching techniques have been proposed to support the retrieval of similar time series based on the time warping distance. A similar subsequence retrieval technique is suggested in [22]. It employs a suffix tree, which has been used extensively as an index structure for substrings search. Substrings can be seen as subsequences in the domain of time series. To enhance the performance, categorization is used to decrease the number of values that the times series can take and thus decrease the number of possible common substrings. A lower bound distance function is devised to filter false dismissals during time series search in the index. This lower bound function would be more effective when time series in databases share many common categories. However, the categorization can introduce error for time sequence values. This leads to more computations in the postprocessing step. Increasing the number of categories gives rise to less error, but this results in the explosion of the number of tree nodes as well as storage space. The results in [22] are illustrated on relatively small databases, with a small average query length of 20.

In [16], the piecewise constant approximation technique is combined with time warping, resulting in the Piecewise Dynamic Time Warping (PDTW) method. PDTW tries to obtain an approximation of the time series by taking the average of every w points of data, where w is determined arbitrarily. A sequence x of length N can be reduced to a shorter sequence $\vec{x_a}$ of average values, $\vec{x_a} = \{a_0, a_1, \ldots, a_n\}$, where $n = \lceil N/w \rceil - 1$. The approximation of the time series is used for obtaining the time warping distance, including the factor of ω to compensate for different lengths of warping paths. We may consider PDTW as a simplified version of the low resolution time warping method that we propose. While both methods obtain an approximation of the original sequence, PDTW does not provide a refined model of distance compensation. We shall see such a distance compensation method in our proposed mechanism in Section 5.

In [32], another strategy for time series search supporting time warping is proposed. In this strategy, two major steps are involved.

1. First, K-L transform is applied to map the original time sequences to lower dimension feature vectors, then a multidimensional index called the Fastmap index is built. If we are looking for a set of sequences $\vec{y}$ within time warping distance $\epsilon_{timewarp}$ of query $\vec{x}$, i.e., $D_{timewarp}(\vec{x}, \vec{y}) \leq \epsilon_{timewarp}$, then the Fastmap index is queried using the same search range $\epsilon_{timewarp}$ with *Euclidean distance function*, i.e.,

$$D(\vec{x}, \vec{y'}) \leq \epsilon_{fastmap} = \epsilon_{timewarp},$$

   where $\vec{y'}$ returns a set of candidate sequences containing some false alarms as well as *false dismissals*. As time warping distance does not satisfy the triangle inequality, any indexing technique which assumes the triangle inequality, cannot avoid producing false dismissals, and so does the Fastmap index. $\epsilon_{timewarp}$ is used as an estimation to the search range of Euclidean distance function D in order to retrieve a smaller set of candidate sequences in the database.
2. Second, a filtering function is proposed to prune away false alarms from the candidate sequences in a postprocessing step. In the filtering process, a **lower bound distance function** D_{lb} that underestimates the time warping distance function is used, that is, $D_{lb}(\vec{x}, \vec{y}) \leq D_{timewarp}(\vec{x}, \vec{y})$. Only sequences whose $D_{lb}(\vec{x}, \vec{y}) \leq \epsilon_{timewarp}$ will be returned.

The resulting sequences after the two steps are then checked by the actual time warping distance computation. Instead of merely using $D_{timewarp}$ ($O(\|\vec{x}\| \times \|\vec{y}\|)$ complexity), D_{lb} (linear complexity) can be used as a filter to quickly prune away nonqualified time series in the candidate sequences set $\vec{y'}$. As a result of the $D_{timewarp}$ underestimation, some false alarms may not be pruned by D_{lb}. The remaining sequences in $\vec{y'}$ are checked against $D_{timewarp}$ to obtain the answer set $\vec{y}$. Experiments in [32] show that this approach can achieve a significant speedup by trading off a small amount of false dismissals. We will analyze this approach in more detail in Section 4.

3 Similarity Search without Time Warping[3]

While time warping is an important feature for applications like voice recognition and similarity matching for financial series, the high complexity deters users from actual deployment. Many research works are based on Euclidean distance matching for stock sequences. In this section, we focus on time series matching without the consideration of time warping. We would first like to define the similarity model used in sequence matching.The first definition is based on the Euclidean distance $D(\vec{x}, \vec{y})$ between time sequences $\vec{x}$ and $\vec{y}$.

Definition 1. *Given a threshold ϵ, two sequences $\vec{x}$ and $\vec{y}$ of equal length n are* **similar** *if*

$$D(\vec{x}, \vec{y}) = \left(\sum_{i=0}^{n-1} (y_i - x_i)^2 \right)^{\frac{1}{2}} \leq \epsilon.$$

A shortcoming of Definition 1 is demonstrated in Fig. 1. From human interpretation, $\vec{x}$ and $\vec{y}$ may be quite similar because $\vec{y}$ can be shifted up vertically to obtain $\vec{x}$ or vice versa. However, they will be considered *not* similar by Definition 1 because errors are accumulated at each pair of x_i and y_i. Therefore, we suggest another similarity model.

Definition 2. *Given a threshold ϵ, two sequences $\vec{x}$ and $\vec{y}$ of equal length n are* **v-shift similar** *if*

$$D(\vec{x}, \vec{y}) = \left(\sum_{i=0}^{n-1} ((y_i - x_i) - (y_A - x_A))^2 \right)^{\frac{1}{2}} \leq \epsilon,$$

where

$$x_A = \frac{1}{n} \sum_{i=0}^{n-1} x_i \quad \text{and} \quad y_A = \frac{1}{n} \sum_{i=0}^{n-1} y_i.$$

3. The work in this section has been reported in [8].

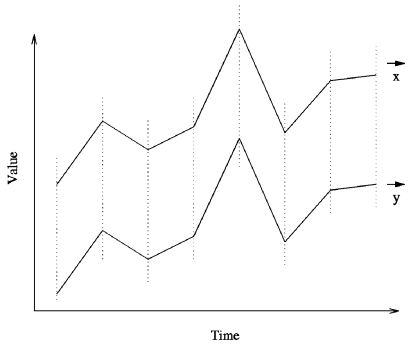


Fig. 1. Example of vertical shifts of time sequences.

From Definition 2, any two time sequences are said to be *v-shift* similar if the Euclidean distance is less than or equal to a threshold ϵ neglecting their vertical offsets from x-axis. This definition can give a better estimation of the similarity between two time sequences with similar trends running at two completely different levels. It can support the requirements in applications such as "pairs trading" for financial data [28].

The Haar wavelets is chosen since it can be computed quickly and easily, requiring linear time in the length of the sequence and simple coding. The complexity of Haar transform can be evaluated by considering the number of operations involved in the recursion process. (Related implementations can be found in [10].) The following is proved in [8].

Lemma 1. *Given a time sequence of length n, where n is an integral power of 2, the complexity of Haar transform is $O(n)$.*

Hence, the complexity of Haar transform is $O(n)$ while $O(n \log n)$ computation is required for Fast Fourier Transform (FFT) [13]. Both impose restriction on the length of time sequences which must be an integral power of 2. Although these computations are all involved in preprocessing stage, the complexity of the transformation can be a concern especially when the database is large.

Apart from Euclidean distance, our model can easily accommodate *v-shift* similarity of two time sequences (Definition 2) at a minor cost. Note that similar to DFT, DWT will not require massive index reorganization because of database updating, which is a major drawback in using the K-L transform or SVD approach.

3.1 Guarantee of no False Dismissal

For FT and DFT, it is shown by Parseval's Theorem [21] that the energy of a signal is conserved in both time and frequency domains. Parseval's theorem also says that the Euclidean distances of time and frequency domains are the same for DFT. This is a very important property in order that dimension reduction of sequence data is possible. It guarantees that no qualified time sequence will be rejected, thus no false dismissal. Here, we show such a relationship for the Haar wavelet transform.

Lemma 2. *Given a sequence $\vec{x} = \{x_0, x_1\}$ and a sequence $\vec{y} = \{y_0, y_1\}$. Suppose the Haar transforms of $\vec{x}$ and $\vec{y}$ are $H(\vec{x}) = \vec{s} = \{s_0, s_1\}$ and $H(\vec{y}) = \vec{r} = \{r_0, r_1\}$, respectively. Lengths of $\vec{x}$, $\vec{y}$, $\vec{s}$, and $\vec{r}$ are all equal to 2. Then, Euclidean distance $D(\vec{x}, \vec{y})$ is $\sqrt{2}$ times of Euclidean distance $D(\vec{s}, \vec{r})$: $D(\vec{x}, \vec{y}) = \sqrt{2} D(\vec{s}, \vec{r})$.*

Proof. Express $\vec{s}$ in terms of $\vec{x}$ and $\vec{r}$ in terms of $\vec{y}$ by applying a similar transformation as in (1) accordingly

$$\vec{s} = \left\{\frac{x_0 + x_1}{2}, \frac{x_0 - x_1}{2}\right\} \qquad \vec{r} = \left\{\frac{y_0 + y_1}{2}, \frac{y_0 - y_1}{2}\right\}.$$

The Square of Euclidean distance of $\vec{s}$ and $\vec{r}$ is

$$D^2(\vec{s}, \vec{r}) = \left(\frac{x_0 + x_1}{2} - \frac{y_0 + y_1}{2}\right)^2 + \left(\frac{x_0 - x_1}{2} - \frac{y_0 - y_1}{2}\right)^2 = \frac{D^2(\vec{x}, \vec{y})}{2}$$

Thus, $D(\vec{x}, \vec{y}) = 2^{\frac{1}{2}} D(\vec{s}, \vec{r})$. □

Lemma 3. *Given two sequences $\vec{x}$ and $\vec{y}$, and the Haar transforms of $\vec{x}$, $\vec{y}$ are $\vec{s}$ and $\vec{r}$, respectively. Lengths of $\vec{x}$, $\vec{y}$, $\vec{s}$ and $\vec{r}$ are all n ($n \geq 2$ and n is a power of 2).*

$$\vec{r} - \vec{s} >= \{C, D_1, D_2, \ldots, D_{n-1}\}.$$

The Euclidean distance $D(\vec{x}, \vec{y}) = S_{\log_2 n}$ can be expressed in terms of $\{C, D_1, D_2, \ldots, D_{n-1}\}$, recursively by

$$\begin{aligned} S_{i+1} &= \sqrt{2} \times (S_i^2 + D_{2^i}^2 + D_{2^i+1}^2 + \cdots + D_{2^{i+1}-1}^2)^{\frac{1}{2}} \quad \text{for } 0 \leq i \leq \log_2 n - 1 \\ S_0 &= C \end{aligned} \tag{3}$$

Proof. In Fig. 2, the original sequence $\vec{x}$ is represented at Level $\log_2 n$. The values of $x_{i,j}$ and d_{2^i+j} are defined by

$$x_{i,j} = \frac{x_{i+1,2j} + x_{i+1,2j+1}}{2} \qquad d_{2^i+j} = \frac{x_{i+1,2j} - x_{i+1,2j+1}}{2}.$$

The Haar transform of $\vec{x}$, $H(\vec{x})$ is represented by

$$\{x_{0,0}, d_1, d_2, \ldots, d_{2^i+j}, d_{2^i+j+1}, \ldots, d_{n-1}\}.$$

A similar hierarchy exists for another sequence $\vec{y}$. Denote $C = x_{0,0} - y_{0,0}$ and

$$D_i = (d_i \text{ of sequence} \vec{x}) - (d_i \text{ of sequence} \vec{y}),$$

where $1 \leq i \leq n - 1$.

We can treat the elements at each horizontal level of the hierarchy to be a data sequence. Hence, the sequence at Level i contains data $\{x_{i,0}, x_{i,1}, \ldots, x_{i,2^i-1}\}$.

Let us define S_i to be

$$S_i = \left\{\sum_{j=0}^{2^i-1} (x_{i,j} - y_{i,j})^2\right\}^{\frac{1}{2}}.$$

S_i can be seen as the Euclidean distance between the data sequences at Level i in the hierarchies for $\vec{x}$ and $\vec{y}$. Also, $S_{\log_2 n}$ is the Euclidean distance between the given time series.

Next we prove the following statement

$$\begin{aligned} S_{i+1} &= \sqrt{2} \times (S_i^2 + D_{2^i}^2 + D_{2^i+1}^2 + \cdots + D_{2^{i+1}-1}^2)^{\frac{1}{2}} \quad \text{for } 0 \leq i \leq \log_2 n - 1 \\ S_0 &= C. \end{aligned}$$

The base case is shown true by Lemma 2 when $i = 0$,

$$S_1 = \sqrt{2} \times (S_0^2 + D_1^2)^{\frac{1}{2}}.$$

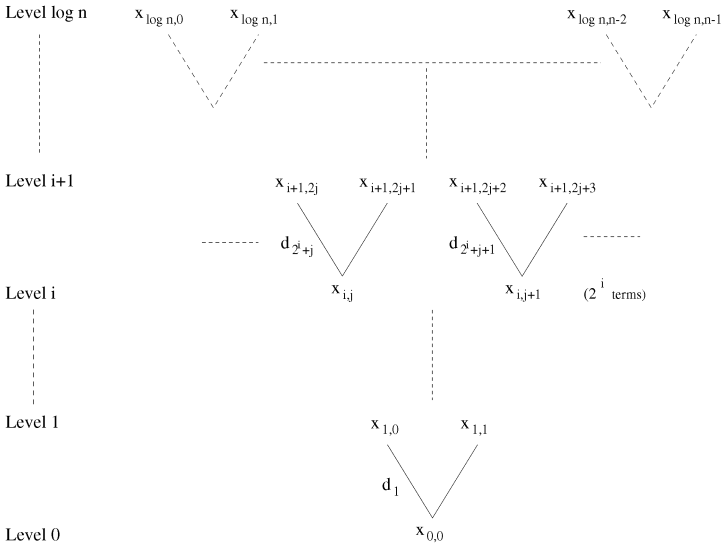


Fig. 2. Hierarchy of Haar wavelet transform of sequence $\vec{x}$ of length n.

We next prove the case for $i = k > 0$. We first note that in the given hierarchy, for a pair of adjacent elements at a level above Level 0 of the form $\{x_{i+1,2j}, x_{i+1,2j+1}\}$, we have the following relation

$$\begin{aligned}&\left(x_{i+1,2j} - y_{i+1,2j}\right)^2 + \left(x_{i+1,2j+1} - y_{i+1,2j+1}\right)^2 = \\ &2\left(\left(x_{i,j} - y_{i,j}\right)^2 + \left(d_{2^i+j} - d'_{2^i+j}\right)^2\right),\end{aligned} \tag{4}$$

where d'_{2^i+j} is the element in the hierarchy for $\vec{y}$ corresponding to d_{2^i+j}. This can be shown by repeating the proof in Lemma 2, replacing $\vec{x}$ by $\{x_{i+1,2j}, x_{i+1,2j+1}\}$, $\vec{y}$ by $\{y_{i+1,2j}, y_{i+1,2j+1}\}$, $\vec{s}$ by $\{x_{i,j}, d_{2^i+j}\}$, and $\vec{r}$ by $\{y_{i,j}, d'_{2^i+j}\}$. Note that $\left(d_{2^i+j} - d'_{2^i+j}\right)^2 = D^2_{2^i+j}$.

For $i = k$,

$$\begin{aligned}S_{k+1} &= \left\{\sum_{j=0}^{2^{k+1}-1} \left(x_{k+1,j} - y_{k+1,j}\right)^2\right\}^{\frac{1}{2}} \\ &= \{\left(x_{k+1,0} - y_{k+1,0}\right)^2 + \left(x_{k+1,1} - y_{k+1,1}\right)^2 \\ &\quad + \cdots + \left(x_{k+1,2^{k+1}-1} - y_{k+1,2^{k+1}-1}\right)^2\}^{\frac{1}{2}}.\end{aligned}$$

By (4), we have

$$\begin{aligned}&S_{k+1} \\ &= \left\{2\left[\left(x_{k,0} - y_{k,0}\right)^2 + D^2_{2^k}\right] + 2\left[\left(x_{k,1} - y_{k,1}\right)^2 + D^2_{2^k+1}\right] + \cdots \right. \\ &\quad \left. + 2\left[\left(x_{k,2^k-1} - y_{k,2^k-1}\right)^2 + D^2_{2^k+2^k-1}\right]\right\}^{\frac{1}{2}} \\ &= \left\{2\left[\left(x_{k,0} - y_{k,0}\right)^2 + \left(x_{k,1} - y_{k,1}\right)^2 + \cdots + \left(x_{k,2^k-1} - y_{k,2^k-1}\right)^2\right]\right. \\ &\quad \left. + 2\left[D^2_{2^k} + D^2_{2^k+1} + \cdots + D^2_{2^k+2^k-1}\right]\right\}^{\frac{1}{2}}.\end{aligned}$$

Finally, by the definition of S_k,

$$S_{k+1} = \sqrt{2} \times \left(S_k^2 + D^2_{2^k} + D^2_{2^k+1} + \cdots + D^2_{2^{k+1}-1}\right)^{\frac{1}{2}}$$

which completes the proof. □

Note that the Haar transform with a coefficient of $1/2$ in (1) does not preserve Euclidean distance. In order to preserve Euclidean distance in the time domains, a normalization is used which effectively alters the coefficient. Given

$$H(\vec{y}) - H(\vec{x}) = \{C, D_1, D_2, D_3, \ldots, D_{2^i-1}\},$$

where $H()$ is computed with the coefficient of $1/2$ in (1), the normalization multiplies $2^{i/2}$ to the coefficient C and D_1; $2^{(i-1)/2}$ to the coefficients D_2, D_3; $2^{(i-2)/2}$ to the coefficients $D_4, D_5, D_6, D_7 \ldots$; and $\sqrt{2}$ to the coefficients $D_{2^{i-1}}, \ldots, D_{2^i-1}$.

For example, with

$$H(\vec{y}) - H(\vec{x}) = \{C, D_1, D_2, D_3, \ldots, D_{15}\},$$

the normalization multiplies 2^2 to the coefficient C and D_1, $2^{3/2}$ to the coefficients D_2, D_3, 2 to the coefficients D_4, D_5, D_6, D_7, and $\sqrt{2}$ to the coefficients $D_8, , \ldots, D_{15}$. In actual implementation, we do not need to do the normalization after subtracting the two values of $H()$. This normalization can be achieved by the scaling factor in (1) from $1/2$ to $1/\sqrt{2}$ in the Haar transformation. Then, we can simply subtract the resulting Haar transformations. With this normalization, Euclidean distance between sequences in the Haar domain will be equivalent to $S_{\log_2 n}$ in (3). The preservation of Euclidean distance of Haar transform ensures the completeness of feature extraction as in DFT. Throughout the remaining discussion in this section, when we compute the Haar transform, we also apply the normalization.

If only the first h_c dimensions ($1 \le h_c \le n$) of Haar transform are used in calculation of Euclidean distance in (3), we should replace 0's in the Haar transformed sequences. This replacement starts from the $(h_c + 1)$th to the nth coefficients in the transformed sequences.

Theorem 1. *Using Haar transform with the above normalization, if the first $h_c (1 \le h_c \le n)$ dimensions of Haar transform are used, no false dismissal will occur for range querying.*

Proof. Considering the inequality in Definition 1 and Lemma 3, if two sequences $\vec{x}$ and $\vec{y}$ are similar, then

$$D(\vec{x}, \vec{y}) = S_{\log_2 n} \leq \epsilon. \quad (5)$$

Let $\vec{s}$ and $\vec{r}$ be the Haar transforms of $\vec{x}$ and $\vec{y}$, respectively. Using the first h_c dimensions of $\vec{s}$ and $\vec{r}$ as index, the value of D_i in (3) will become zero for $i \geq h_c$. Thus, if $\vec{r}$ and $\vec{s}$ are both truncated in this manner, the Euclidean distance between the two truncated sequences is $\leq S_{\log_2 n} \leq \epsilon$. This completes the proof. □

3.2 The Overall Strategy

In this section, we present the overall strategy for handling time series querying based on Euclidean distances. We shall need preprocessing to extract the feature vectors with reduced dimensionality, and to build the index. After the index is built, content-based search can be performed for two types of querying: range querying and n-nearest-neighbors querying.

Preprocessing:

Step 1—Similarity Model Selection. According to their applications users may choose to use either the simple Euclidean distance (Definition 1) or the v-shift similarity (Definition 2) as their similarity measurements. For Definition 1, Haar transform is applied to time series. For Definition 2, Haar transform is applied, but the first Haar coefficient will not be used in indexing, as there is no need to match their average values anymore.

Step 2—Index Construction. Given a database of time series of varying length. We preprocess the time series as follows: We obtain the ω-point Haar transform by applying (1) with the normalization factor $1/\sqrt{2}$, to each subsequence with a sliding window of size ω for each sequence in the database. An index structure such as an R-Tree is built, using the first h_c[4] Haar coefficients where h_c is an optimal value found by experiments based on the number of page accesses. This is because of a trade off between postprocessing cost and index dimension.

Range Query. After we have built the index, we can carry out range query or nearest neighbor query evaluation for query sequences of length ω. For range queries, two steps are involved: 1) Similar sequences with distances $\leq \epsilon$ from the query are looked up in the index and returned and 2) A postprocessing step is applied to these sequences to obtain the actual distances in time domain to remove all false alarms.

Nearest Neighbor Query. For n-nearest neighbor query, we propose a two-phase evaluation.

- *Phase 1.* In the first phase, n nearest-neighbors of query $\vec{q}$ are found in the R-Tree index using the algorithm in [25]. The Euclidean distances D in time domain (full dimension) between the transformed query sequence and all the n nearest neighbors are obtained, which are given by $D(\vec{q}, \vec{nn_i^1})$, where $\vec{nn_i^1}$ denotes the ith nearest neighbor $(1 \leq i \leq n)$, with $\vec{nn_n^1}$ farthest from the query $\vec{q}$.
- *Phase 2.* A range query evaluation is then performed on the same index by setting $\epsilon = D(\vec{q}, \vec{nn_n^1})$ initially. During the search, we keep a list of n nearest sequences $\vec{nn_i^2}$ found so far and their Euclidean distances in time domain (full dimension) $D(\vec{q}, \vec{nn_i^2})$ with query $\vec{q}(1 \leq i \leq n)$. The postprocessing step as used for range query is avoided since the Euclidean distances are already found in time domain during the search. During the search we keep updating the value of ϵ by $D(\vec{q}, \vec{nn_n^2})$. The n-nearest neighbors stored in the list are returned as answer when the range query evaluation is finished. Let $D(\vec{q}, \vec{nn_n^{ans}})$ be the distance of the farthest nearest neighbor with query $\vec{q}$.

The effectiveness of this n-nearest neighbor search algorithm arises from the value of $D(\vec{q}, \vec{nn_n^1})$ found in Phase 1 which provides a sufficient and small query range to prune away a large amount of candidates in Phase 2. No false dismissal will occur in Phase 2 as $D(\vec{q}, \vec{nn_n^1})$ gives the upper bound distance for $D(\vec{q}, \vec{nn_n^{ans}})$ which is the farthest n nearest neighbor in the final answer. The extra step introduced in Phase 2 to update ϵ can enhance the performance by pruning more nonqualified minimum bounding rectangles (MBRs) during the traversal of R-Tree.

3.3 Performance Evaluation

Experiments using real stock data and synthetic random walk data have been carried out. In the experiments, we compare the querying performance between the Haar wavelets approach and scaled DFT [30]. For interest of space the detailed results are not shown. We discover that the performance of the two methods are quite similar in terms of precision. We pointed out earlier that the preprocessing time for Haar wavelets is expected to be much less than that for DFT, this expectation is confirmed by the experiments. We conclude that Haar wavelets is a competitive method in the application of time series similarity search.

4 Time Warping

Time (x-axis) shifts of sequences can be coped with by means of time warping techniques [23], [6]. Time warping is widely used in speech and word recognition fields, in which human speech consists of varying durations and paces. In the financial sector, time warping technique offers the same benefits in stock series matching. One is interested in looking up sequences with similar rise and fall patterns. Similar stock patterns in different scales or occurring in different period would be considered similar. This is why time warping distance is sometimes more appropriate for stock sequences matching than Euclidean distance.

The definition of time warping distance is given in Section 2.3. An algorithm for computing time warping distances by dynamic programming [32] is shown in Fig. 3. We consider two sequences

$$\vec{x} = \{x_0, x_1, \ldots, x_{x_{len}-1}\}$$

and

$$\vec{y} = \{y_0, y_1, \ldots, y_{y_{len}-1}\}.$$

$\|\vec{x}\|$ denotes the length of $\vec{x}$. $C_{matrix}[i][j]$ stores the shortest cumulative time-warping distance from $\{x_0, \ldots, x_i\}$ to

4. Using Definition 2, one dimension can be saved in the index tree which corresponds to the first coefficient in the Haar transform.

Algorithm 1 TimeWarpDistance($\vec{x}$,$\vec{y}$)

1 $x_{len} = \|\vec{x}\|$; $y_{len} = \|\vec{y}\|$;
2 $C_{matrix}[0][0] = 0.0$;
3 for $(0 \leq \mathrm{i} \leq x_{len} - 1)$ $C_{matrix}[\mathrm{i}][0] = \infty$;
4 for $(0 \leq \mathrm{j} \leq y_{len} - 1)$ $C_{matrix}[0][\mathrm{j}] = \infty$;
5 for $(0 \leq \mathrm{i} \leq x_{len} - 1)$
6 for $(0 \leq \mathrm{j} \leq y_{len} - 1)$
7 $C_{matrix}[\mathrm{i}][\mathrm{j}] = \{(D(x_i, y_j))^2 + (\min(C_{matrix}[i-1][j], C_{matrix}[i][j-1], C_{matrix}[i-1][j-1]))^2\}^{\frac{1}{2}}$;
8 return $C_{matrix}[x_{len}][y_{len}]$;

Fig. 3. Algorithm for finding time warping distance.

$\{y_0, \ldots, y_j\}$, which is in terms of the Euclidean distance of pair (x_i, y_j), and the minimum among $C_{matrix}[i-1][j]$, $C_{matrix}[i][j-1]$, and $C_{matrix}[i-1][j-1]$ (Line 7). The general principle of the algorithm is to find the shortest cumulative distance for each pair of time values between sequences $\vec{x}$ and $\vec{y}$, starting from the first pair (x_0, y_0), till the last pair $(x_{x_{len}-1}, y_{y_{len}-1})$. The result is the shortest cumulative distance $C_{matrix}[x_{len} - 1][y_{len} - 1]$.

Example 1. Consider two sequences $\vec{x} = \{4, 3, 1, 2, 3, 0\}$ and $\vec{y} = \{1, 2, 0, -1, 1, 2\}$, the corresponding cumulative distance matrix can be found using the algorithm in Fig. 3 and the process is shown in Fig. 4. In Fig. 4, each box corresponds to an entry in the cumulative distance matrix $C_{matrix}[i][j]$. Those pairs constituting the shortest overall cumulative distance are in gray, and the time warping distance is found to be $D_{timewarp}(\vec{x}, \vec{y}) = 4.2$ (upper right corner of the matrix).

We call a path such as the one traced out by the gray units in Fig. 4 a **time warping path**. Though time warping technique can accommodate time shifts of sequences, there are two major difficulties in using time warping distance. First, for length n sequences, the complexity of time warping distance function is $O(n^2)$ (strictly speaking, the complexity should be $O(\|\vec{x}\| \times \|\vec{y}\|)$), as revealed from the distance matrix calculation, compared with $O(n)$ of Euclidean distance matching. This hinders the use of time warping distance for similarity searching in very large time series databases where response time is a critical issue. Second, we cannot directly apply indexing techniques for

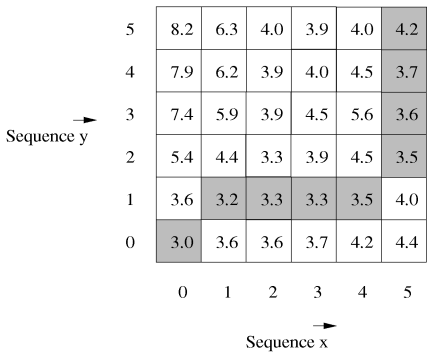


Fig. 4. Cumulative distance matrix C_{matrix} for template and data time series.

time warping distance as in [11], [2] to speed up sequences retrieval. For multidimensional index trees like the R-Tree, the distance function under consideration is assumed to be a *metric*, and time warping distance fails to fulfill this requirement.

Given a nonempty set $\mathcal{X}$, a metric D on $\mathcal{X}$ is a function which assigns to each pair of points a nonnegative real number satisfying the following for all $x, y \in \mathcal{X}$:

1. $D(x, y) \geq 0$ and $D(x, y) = 0$ if and only if $x = y$;
2. $D(x, y) = D(y, x)$;
3. For all $x, y, z \in \mathcal{X}$, $D(x, y) \leq D(x, z) + D(y, z)$, (triangle inequality).

The pair $(\mathcal{X}, D)$ is called a **metric space**[12].

The Euclidean distance is a metric space distance function. We can show that time warping distance violates the triangle inequality by the following example.

Example 2. Given three time sequences

$$\vec{x} = \{0, 1, 3\}, \vec{y} = \{3, 2, 2\},$$

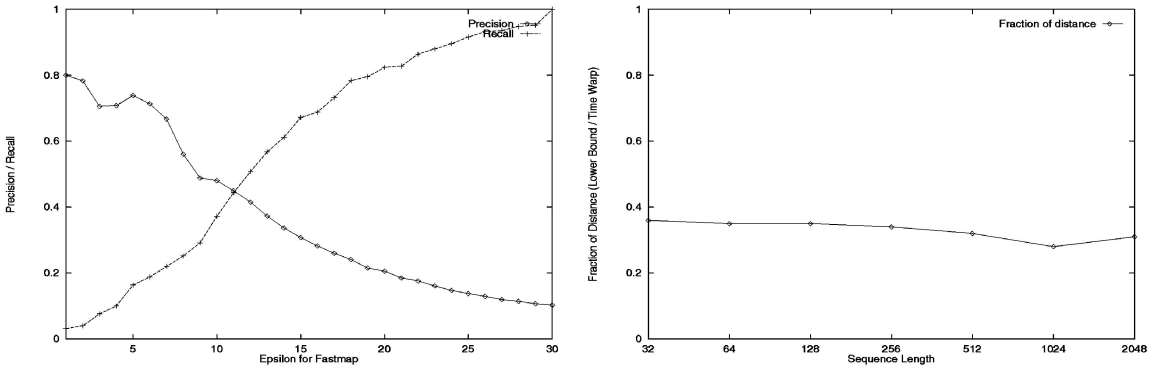
and $\vec{z} = \{2, 3, 2\}$.

$$D_{timewarp}(\vec{x}, \vec{y}) = 11 > D_{timewarp}(\vec{x}, \vec{z}) + D_{timewarp}(\vec{y}, \vec{z}) = 6 + 1 = 7$$

Therefore, if multidimensional or metric index trees are employed for direct indexing based on time warping distance, it may give rise to *false dismissals*. To deal with the problem of time complexity, we propose an approximation function to time warping distance, which results in less computation by trading off a small amount of accuracy. Our proposed method has a similar flavor as the index-based similarity search mechanism proposed in [32].

4.1 Similarity Search Based on K-L Transform and a Lower Bound Function

An approach [32] that employs K-L transform and a lower bound distance function to support range search with time warping distance is described in Section 2.3. The two main steps are 1) K-L transformed sequences are inserted into an index tree (Fastmap index). The time warping range is used as the parameter in a range search on the tree and 2) The postprocessing step removes false alarms. In this step, a lower bound distance function is used to help filter false alarms more efficiently.



(a) (b)

Fig. 5. (a) Precision and recall of Fastmap. (b) Fraction estimated by D_{lb}.

The main difficulty of this technique is that the search range of time warping distance, $\epsilon_{timewarp}$, when used in extracting sequences in the Fastmap index, is not effective in finding a relatively small candidate set with little false dismissals. (In fact, the value of $\epsilon_{fastmap}$ is increased to $\epsilon^2_{timewarp}$ in [32] to reduce the number of false dismissals.) Therefore, large amounts of sequences have to be checked with the lower bound distance function D_{lb} (described in Section 2.3) and then the time warping distance function $D_{timewarp}$ is used to obtain the answer.

We have conducted experiments based on precision and recall using $\epsilon_{fastmap}$ as the search range for the Fastmap index (excluding the pruning step by D_{lb}). Precision and recall are defined as follows:

$$\begin{aligned} \text{Precision} &= \frac{S_{RetrievedAndQualified}}{S_{RetrievedTWrecall}} \\ \text{Recall} \quad &= \frac{S_{RetrievedAndQualified}}{S_{Qualified}}, \end{aligned} \tag{6}$$

where $S_{Retrieved}$ is the number of sequences retrieved from Fastmap index, $S_{Qualified}$ is the number of sequences qualified, and $S_{RetrievedAndQualified}$ is the number of sequences retrieved and qualified.

Totally 50 sequences are picked out and queried on a database of 5k synthetic random walk sequences of length 256. The value of $\epsilon_{timewarp}$ is fixed such that the number of qualified sequences in the result set is 2.75 percent of the database size. The value of $\epsilon_{fastmap}$ is varied such that at least one qualified sequence is retrieved and $\epsilon_{fastmap}$ will be increased until all qualified sequences are retrieved. The result is shown in Fig. 5a. It is observed that when we want to have higher recall, the value of $\epsilon_{fastmap}$ should be increased. However, precision drops rapidly with $\epsilon_{fastmap}$, which results in large amounts of false alarms. The consequence is that more processing time is devoted to the matching of candidate sequences with D_{lb} and $D_{timewarp}$. As the complexity of $D_{timewarp}$ is $O(n^2)$, the performance drops drastically with the increase in length of time series, which is confirmed in [32].

D_{lb} underestimates $D_{timewarp}$ to a great extent. For the same time series database we use, the fraction of distance estimated by D_{lb} is shown in Fig. 5b. The pruning power of D_{lb} is low since only 25 percent to 35 percent of the time warping distance on the average can be estimated. Therefore large amounts of remaining sequences still should be checked with $D_{timewarp}$, which involves the most computations.

5 Low Resolution Time Warping

We try to improve the overall performance of the range querying with Fastmap index by replacing the lower bound distance function as described in Section 4.1 with our proposed approximation function as a more effective filter in the postprocessing step. This function is capable of pruning a larger number of false alarms arising from large $\epsilon_{fastmap}$. The proposed approximation function is called the **Low Resolution Time Warping**, denoted by $D_{lowresTW}$ (see (7)). It can act as both an approximation function and a filtering function. To obtain low resolution time warping, two steps are involved, resolution reduction and distance compensation.

5.1 Resolution Reduction of Sequences

To arrive at different resolutions of sequences, we employ multiresolution representation of the Haar wavelets. Upon application of Haar transform on time sequences, Haar coefficients can be obtained. We call this process *decomposition*. Conversely, we may also reconstruct time sequences by applying an inverse Haar transform to Haar coefficients. Recall that the Haar transformation is described in (1). (In time-warping considerations, we assume the normalization factor of $1/2$ as in (1) as we do not have the problem of preserving Euclidean distances.) The inverse Haar transformation, or *reconstruction* goes in a similar but reverse manner. It is shown in (7),

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x'_0 \\ d_{1,0} \\ x'_1 \\ d_{1,1} \end{bmatrix} \tag{7}$$

with Haar coefficients $\vec{w} = [\, x'_0 \quad d_{1,0} \quad x'_1 \quad d_{1,1} \,]^T$ as input and $\vec{x} = [x_0 \quad x_1 \quad x_2 \quad x_3\,]^T$ as output. The number of iterations for recovering the original sequence in the reconstruction is exactly the same as that needed in decomposition. Different resolutions of time series can be achieved by varying the

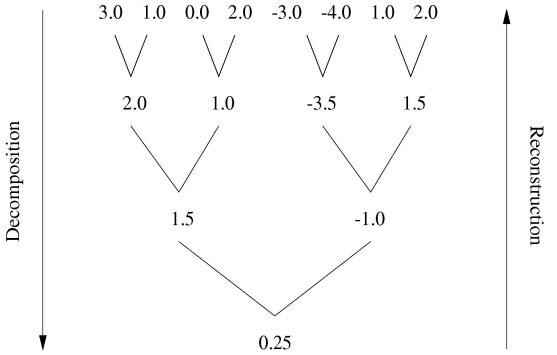


Fig. 6. Resolution reduction by variations of iterations.

number of iterations performed in (7) of the reconstruction process. The more the iterations, the higher the resolution we can obtain.

In Fig. 6, we show different resolutions of an input sequence $\vec{z} = \{3, 1, 0, 2, -3, -4, 1, 2\}$. Decomposition and reconstruction correspond to downward and upward traversals of the tree respectively, with upper levels representing higher resolutions. By the reconstruction process, we can obtain four different resolutions which are

$$\{0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25\},$$
$$\{1.5, 1.5, 1.5, 1.5, -1.0, -1.0, -1.0, -1.0\},$$
$$\{2.0, 2.0, 1.0, 1.0, -3.5, -3.5, 1.5, 1.5\},$$

and $\vec{z}$ itself. Note that the lengths of sequences are preserved at different resolutions.

With the above method, for a sequence of length 256, there are only eight resolutions available. We can get many more different resolutions by varying the resolutions of sequence segments. For instance, we can obtain a finer resolution of sequence $\{2.0, 2.0, 1.0, 1.0, -3.5, -3.5, 1.5, 1.5\}$ by expanding the last two values $\{1.5, 1.5\}$ to $\{1.0, 2.0\}$, which becomes $\{2.0, 2.0, 1.0, 1.0, -3.5, -3.5, 1.0, 2.0\}$. A systematic way to achieve this variety of resolutions is illustrated with an example below (see the left-hand side of Fig. 7).

In the above example, the Haar transformed sequence is $H(\vec{x}) = \{0.25, 1.25, 0.5, -2.5, 1.0, -1.0, 0.5, -0.5\}$. We obtain different resolutions of $\vec{z}$ by first replacing some Haar coefficients $H(\vec{z})$ by zero, then performing a full reconstruction (a full iteration of matrix multiplication) of (7). For example, the bottom sequence of

$$\{0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25\}$$

is obtaned by first replacing by zero all elements except the first one in $H(\vec{x})$, giving $H' = \{0.25, 0, 0, 0, 0, 0, 0, 0\}$, then reconstructing the sequence from H'. The number of coefficients replaced by zero determines the resolution of sequence $\vec{z}$. Possible resolutions of sequence $\vec{z}$ are shown in Fig. 7, with no replacement at Level 0, and seven coefficients replaced at Level 7.

Although we obtain lower resolution versions of a sequence, their lengths are still equal to the original sequence length. This does not constitute any improvement on time complexity. Therefore, *sampling* is introduced for sequence length reduction. On the left-hand side of Fig. 7, there exists some consecutive repeated values in different resolutions of sequences. One sample value can thus be taken out of those repeated ones. Consider the same figure, the sequence at Level 2 is sampled from

$$\{3.0, 1.0, 0.0, 2.0, -3.5, -3.5, 1.5, 1.5\}$$

to $\{3.0, 1.0, 0.0, 2.0, -3.5, 1.5\}$, with the last two value pairs sampled down to one value each. By the same rule, the sampled sequence at the last level will be of length 1 which is $\{0.25\}$. The right column of Fig. 7 shows the result after sampling of sequences on the left. The underlining signifies the entries that are sampled. Let us call this process of removing duplicates as **down sampling**.

5.2 Distance Compensation

With reduced sequence length, the computations involved in time warping distance can be drastically reduced. However, it is still inappropriate to estimate the original distance by the distance of lower resolution sequences, as distances that arise from discarded value pairs are lost owing to the removal of duplicates by the down sampling process. This may lead to severe underestimation of the original time warping distance. The aim of using lower resolution sequences, down sampling, and then time warping is to first pair up as rapidly as possible the peaks and valleys of the two sequences accordingly. Afterwards, we should compensate for the distances lost owing to the down sampling process.

We illustrate the compensation process with an example. Given two sequences $\vec{x}$ and $\vec{y}$ of length 4. (Note that, in general, lengths of $\vec{x}$ and $\vec{y}$ need not be the same.) Without loss of generality, we denote their low resolution versions as $\vec{X} = \{X_0, X_1, X_2, X_3\}$ and $\vec{Y} = \{Y_0, Y_1, Y_2, Y_3\}$, and assume that they have a time warping path shown in gray on the distance matrix in Fig. 8. The length of the path corresponds to the number of pairs of matching values in time warping, which is five in this case, namely, (X_0, Y_0), (X_1, Y_1), (X_2, Y_1), (X_3, Y_2), and (X_3, Y_3). We say that each matching pair correspond to a **state**. The states are visualized in the lower part of Fig. 8. There are five different states, with arrows pointing to the matched pairs. The positions of these arrows are able to indicate where and how we make the distance compensation.

There are two possible movements of the arrow pair when proceeding from one state to the next: either one arrow moves a unit forward, or both arrows move a unit forward. For any consecutive elements $\mathbf{X_i}$, $\mathbf{X_{i+1}}$ and $\mathbf{Y_j}$, $\mathbf{Y_{j+1}}$ of sequences $\vec{X}$ and $\vec{Y}$, respectively, where i need not equal j, there may exist some repeated values or elements which are in-between (removed by down sampling), represented as X_i and Y_j. Denote the arrows pointing at sequences $\vec{X}$ and $\vec{Y}$ as X_{arrow} and Y_{arrow}, respectively. There are two cases.

- *Case 1.* Either X_{arrow} or Y_{arrow} moves forward. Without loss of generality, we assume Y_{arrow} moves a step forward pointing at $\mathbf{Y_{j+1}}$. As the only movement is Y_{arrow}, we just need to compensate for distances between point $\mathbf{X_i}$ and subsequence $\{Y_j, \ldots, Y_j\}$, which is

$$DC = \left\{(\mathbf{X_i} - Y_j)^2 \times |\{Y_j, \ldots, Y_j\}|\right\}^{\frac{1}{2}}$$

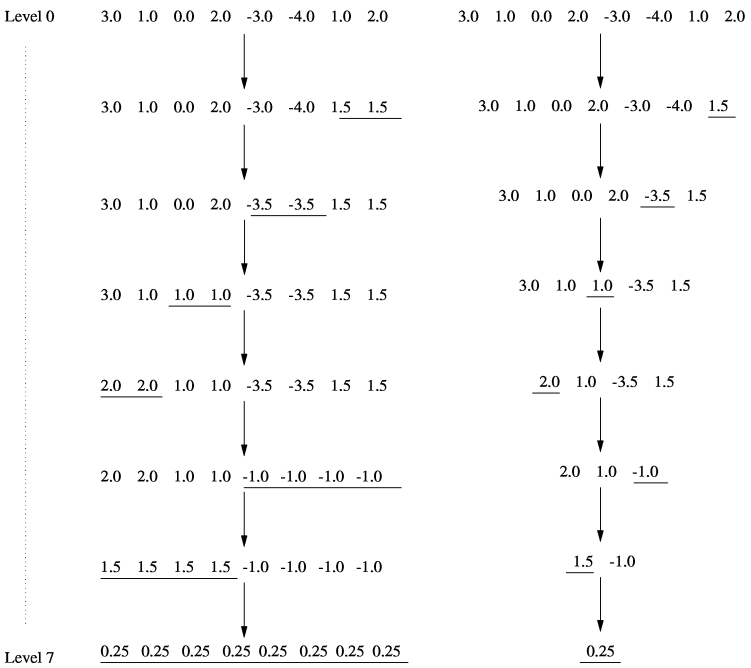


Fig. 7. Resolution reduction and sampling of Haar coeffiecients.

such that the transition of Y_{arrow} from $\mathbf{Y_j}$ to $\mathbf{Y_{j+1}}$ is in continuity. The length of subsequence $\{Y_j, \ldots, Y_j\}$ is equal to the number of repeated values removed for $\mathbf{Y_j}$ in down sampling.

- *Case 2.* Both X_{arrow} and Y_{arrow} move a unit forward. Distances from both subsequences $\{X_i, \ldots, X_i\}$ and $\{Y_j, \ldots, Y_j\}$ should be considered. Euclidean distance between $\{X_i, \ldots, X_i\}$ and $\{Y_j, \ldots, Y_j\}$ could be used, however, a better estimation involves the use of time warping distance. It is computed as follows:

$$\begin{aligned} DC = &\{D^2_{timewarp}(\{\mathbf{X_i}, X_i, \ldots, X_i, \mathbf{X_{i+1}}\}, \\ &\{\mathbf{Y_j}, Y_j, \ldots, Y_j, \mathbf{Y_{j+1}}\}) \\ &- (\mathbf{X_i} - \mathbf{Y_j})^2 - (\mathbf{X_{i+1}} - \mathbf{Y_{j+1}})^2\}^{\frac{1}{2}}. \end{aligned}$$

Knowing how to compensate distance leads us to the formula for finding the overall low resolution time warping distance between two time series $\vec{x}$ and $\vec{y}$ which is shown in the following equation:

$$D_{lowresTW}(\vec{x}, \vec{y}) = \left\{ D^2_{timewarp}(\vec{X}, \vec{Y}) + \sum_{s=1}^{No.ofstates} DC_s^2 \right\}^{\frac{1}{2}}, \quad (8)$$

where $\vec{X}$ and $\vec{Y}$ are the lower resolution versions of sequences $\vec{x}$ and $\vec{y}$, respectively, and DC_s is the distance compensated at state s. We can thus use $D_{lowresTW}$ to closely approximate $D_{timewarp}$.

The procedures in finding low resolution time warping distance are summarized as follows: The time series $\vec{x}$ and $\vec{y}$ go through a resolution reduction process, in which we perform the steps:

1. Haar decomposition,
2. Haar reconstruction,
3. down sampling, and
4. time warping and distance compensation are carried out on the resulting vectors.

This results in the low resolution time warping distance $D_{lowres}TW$.

5.2.1 Time Warping Distance in Case 2

In this section, we discuss in more details how the time-warping distance computation in Case 2 in Section 5.2 can be much simplified. Consider the time warping distance between $\vec{X_s} = \{X_i, \ldots, X_i, X_{i+1}\}$ and

$$\vec{Y_s} = \{Y_j, \ldots, Y_j, Y_{j+1}\}.$$

Let p be the number of X_i in $\vec{X_s}$ and q be the number of Y_j in $\vec{Y_s}$. Let $D(X_i, Y_j) = a$, $D(X_{i+1}, Y_j) = b$, $D(X_i, Y_{j+1}) = c$, and $D(X_{i+1}, Y_{j+1}) = d$, we have a, b, c, and $d \geq 0$. The basic idea can be shown by two simple examples. The following tables show the derivation of the $D_{timewarp}$ between $\vec{X_s}$ and $\vec{Y_s}$ for two different cases: 1) $p = 3, q = 6$ and 2) $p = 6, q = 3$. They are the cumulative distance matrices similar to the one shown in Fig. 4. From Table 1, we generalize the case for $p < q$:

$$\begin{aligned} &if\ a \leq b, Z = \sqrt{qa^2 + d^2}; \\ &if\ a \geq b, Z = \sqrt{pa^2 + (q-p)b^2 + d^2}. \end{aligned}$$

Similarly, from Table 2, for $p > q$:

$$\begin{aligned} &if\ a \leq c, Z = \sqrt{pa^2 + d^2}; \\ &if\ a \geq c, Z = \sqrt{qa^2 + (p-q)c^2 + d^2}. \end{aligned}$$

Lemma 4. *Assume the above definitions for $\vec{X_s}, \vec{Y_s}, p, q, a, b, c, d$. If $p \leq q$, we have*

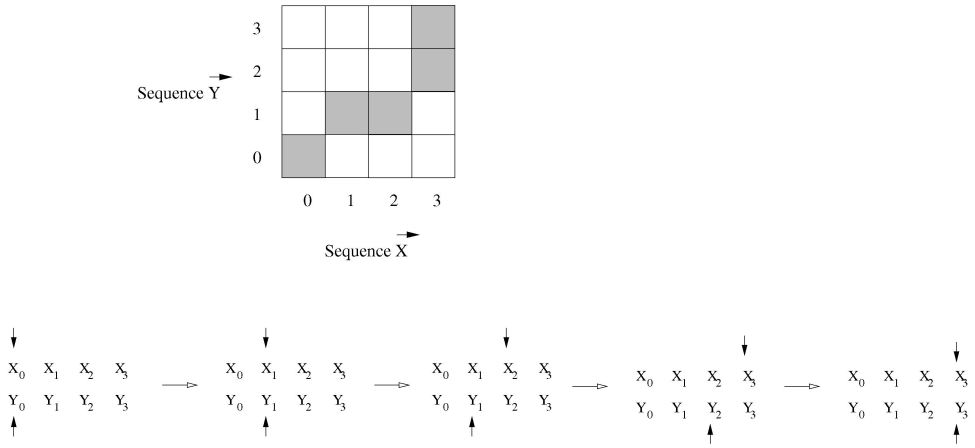


Fig. 8. State transitions in finding time warping distance.

$$D_{timewarp}(\vec{X}_s, \vec{Y}_s) = \begin{cases} (pa^2 + (q-p)b^2 + d^2)^{\frac{1}{2}} & \text{if } a \geq b \\ (qa^2 + d^2)^{\frac{1}{2}} & \text{if } a \leq b. \end{cases} \quad (9)$$

If $p \geq q$, we have

$$D_{timewarp}(\vec{X}_s, \vec{Y}_s) = \begin{cases} (qa^2 + (p-q)c^2 + d^2)^{\frac{1}{2}} & \text{if } a \geq c \\ (pa^2 + d^2)^{\frac{1}{2}} & \text{if } a \leq c. \end{cases} \quad (10)$$

The above lemma can be easily deduced. By employing (9) and (10), the complexity for Case 2 compensation can be reduced from quadratic for general time-warping distance computation to linear.

5.3 An Example

To obtain the low resolution time warping distance between two sequences $\vec{x} = \{3, 1, 0, 2, -3, -4, 1, 2\}$ and $\vec{y} = \{2, 3, 1, -3, -4, 1, 0, 1\}$ of length $n = 8$, the following procedures are taken. First we obtain the lower resolution versions of both $\vec{x}$ and $\vec{y}$ by finding their Haar coefficients using (1) repeatedly, which are

$$H(\vec{x}) = \{0.25, 1.25, 0.5, -2.5, 1.0, -1.0, 0.5, -0.5\}$$

and

$$H(\vec{y}) = \{0.125, 0.625, 1.75, -1.0, -0.5, 2.0, -2.5, -0.5\},$$

respectively. If we choose to use three sample values, we replace the $(8-3) = 5$ rightmost Haar coefficients each in $H(\vec{x})$ and $H(\vec{y})$ by zeros, resulting in

$$H(\vec{x}) = \{0.25, 1.25, 0.5, 0, 0, 0, 0, 0\}$$

and $H(\vec{y}) = \{0.125, 0.625, 1.75, 0, 0, 0, 0, 0\}$. Haar reconstructions are then performed using the modified $H(\vec{x})$ and $H(\vec{y})$ to obtain lower resolution sequences of $\vec{x}$ and $\vec{y}$, which are

$$\{2.0, 2.0, 1.0, 1.0, -1.0, -1.0, -1.0, -1.0\}$$

and

$$\{2.5, 2.5, -1.0, -1.0, -0.5, -0.5, -0.5, -0.5\},$$

respectively. They are then reduced to $\vec{X} = \{2.0, 1.0, -1.0\}$ and $\vec{Y} = \{2.5, -1.0, -0.5\}$ after sampling of duplicated time values.

Next, we find the value of $D_{timewarp}(\vec{X}, \vec{Y})$ which is 1.66, with time warping path (X_0, Y_0), (X_1, Y_0), (X_2, Y_1), and (X_2, Y_2). After this, we perform the distance compensation. According to this path, two Case 1 and one Case 2 distance compensations are required. The last pair of time values (X_2, Y_2) also needs compensation as duplicated values follow after them, i.e., $\{-1.0, -1.0, -1.0, -1.0\}$ of $\vec{x}$ and $\{-0.5, -0.5, -0.5, -0.5\}$ of $\vec{y}$. The procedure for this compensation is nearly the same as for those described in Case 2.

TABLE 1
Cumulative Distance Matrix for Sequences $\vec{X}_s$ and $\vec{Y}_s$ $(p = 3, q = 6)$

Y_{j+1}	$\sqrt{6a^2 + c^2}$	$\sqrt{6a^2 + c^2}$	$\sqrt{6a^2 + c^2}$	Z
Y_j	$\sqrt{6}a$	$\sqrt{6}a$	$\sqrt{6}a$	$\min\{\sqrt{5a^2 + b^2}, \sqrt{3a^2 + 3b^2}\}$
Y_j	$\sqrt{5}a$	$\sqrt{5}a$	$\sqrt{5}a$	$\min\{\sqrt{4a^2 + b^2}, \sqrt{3a^2 + 2b^2}\}$
Y_j	$\sqrt{4}a$	$\sqrt{4}a$	$\sqrt{4}a$	$\sqrt{3a^2 + b^2}$
Y_j	$\sqrt{3}a$	$\sqrt{3}a$	$\sqrt{3}a$	$\sqrt{3a^2 + b^2}$
Y_j	$\sqrt{2}a$	$\sqrt{2}a$	$\sqrt{3}a$	$\sqrt{3a^2 + b^2}$
Y_j	a	$\sqrt{2}a$	$\sqrt{3}a$	$\sqrt{3a^2 + b^2}$
	X_i	X_i	X_i	X_{i+1}

$$Z = \min\{\sqrt{6a^2 + d^2}, \sqrt{5a^2 + b^2 + d^2}, \sqrt{3a^2 + 3b^2 + d^2}\}$$

TABLE 2
Cumulative Distance Matrix for Sequences $\vec{X}_s$ and $\vec{Y}_s$ $(p = 6, q = 3)$

Y_{j+1}	$\sqrt{3a^2 + c^2}$	$\sqrt{3a^2 + c^2}$	$\sqrt{3a^2 + c^2}$	$\sqrt{3a^2 + c^2}$	$\min\{\sqrt{3a^2 + 2c^2}, \sqrt{4a^2 + c^2}\}$	$\min\{\sqrt{3a^2 + 3c^2}, \sqrt{5a^2 + c^2}\}$	Z
Y_j	$\sqrt{3}a$	$\sqrt{3}a$	$\sqrt{3}a$	$\sqrt{4}a$	$\sqrt{5}a$	$\sqrt{6}a$	$\sqrt{6a^2 + b^2}$
Y_j	$\sqrt{2}a$	$\sqrt{2}a$	$\sqrt{3}a$	$\sqrt{4}a$	$\sqrt{5}a$	$\sqrt{6}a$	$\sqrt{6a^2 + b^2}$
Y_j	a	$\sqrt{2}a$	$\sqrt{3}a$	$\sqrt{4}a$	$\sqrt{5}a$	$\sqrt{6}a$	$\sqrt{6a^2 + b^2}$
	X_i	X_i	X_i	X_i	X_i	X_i	X_{i+1}

$$Z = \min\{\sqrt{3a^2 + 3c^2 + d^2}, \sqrt{5a^2 + c^2 + d^2}, \sqrt{6a^2 + d^2}\}$$

Finally, by substituting $D_{timewarp}(\vec{X}, \vec{Y}) = 1.66, DC_1 = 0.5$, $DC_2 = 1.5$, $DC_3 = 0$, and $DC_4 = 0.87$ into (8), we obtain

$$D_{lowresTW} = \left\{1.66^2 + 0.5^2 + 1.5^2 + 0.87^2\right\}^{\frac{1}{2}} = 2.45.$$

6 Performance Evaluation

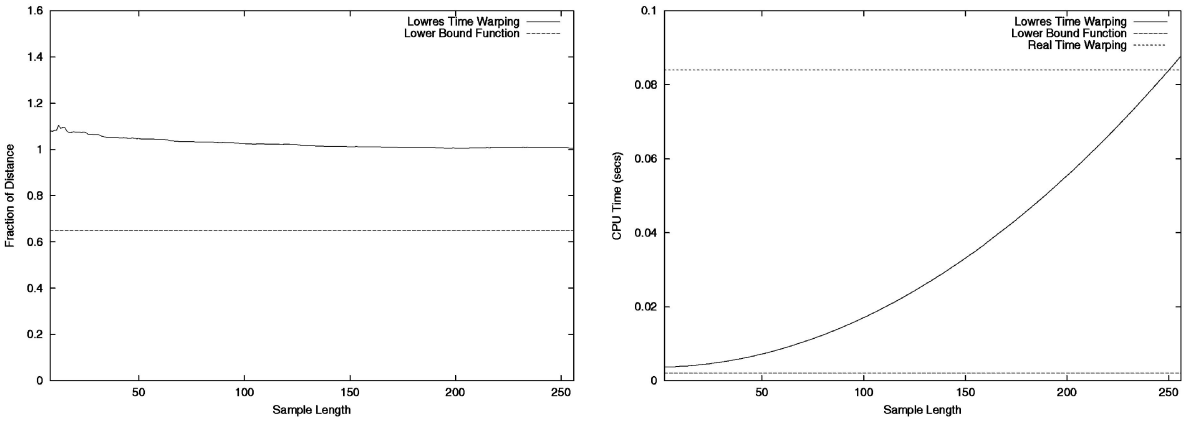
Experiments using both real and synthetic data have been carried out. All experiments are conducted on a Sun UltraSPARC-1 workstation with 592MBytes of main memory. All programs are written in C. Real stock sequences are extracted from the equities of the Hong Kong Stock Market during the period 7/93 - 11/96. It is believed that only *brown noise* or *random walks* exists in real signals. In particular, stock movements and exchange rates can be modeled successfully as random walks in [9], for which a skewed energy spectrum can be obtained. Therefore, synthetic random walk time sequences are also generated for the experiments. The way to generate the random walk data is similar to a number of previous work including [2], [24]. Both the real and synthetic data sets consist of 5k of time sequences. The length of sequences ranges from 64 to 512. The average value of each sequence is normalized to zero. Queries are randomly picked from the data sets as in [31]. Each resulting measurement we report is obtained by taking the average of 50 trials. In the figures showing experimental results, we shall label a figure with (Real) if the experiment is conducted on real data set and (Synthetic) if it is on synthetic data set.

6.1 Accuracy versus Runtime

We aim to show the accuracy and effectiveness of the low resolution approach for time warping approximation. A range query is posed and the answer is obtained by following an exhaustive search of all time series in the database. Both the lower bound function and the low resolution time warping distances are computed. Results of both real and synthetic data are shown in Figs. 9, 10, 11, 12, and 13. In Figs. 9a and 10a, the fraction of estimated distance to the real-time warping against sample length is investigated (original length of time series is 256). **Sample length** refers to the number of sample values taken in down sampling, we denote it as k_{sam}.

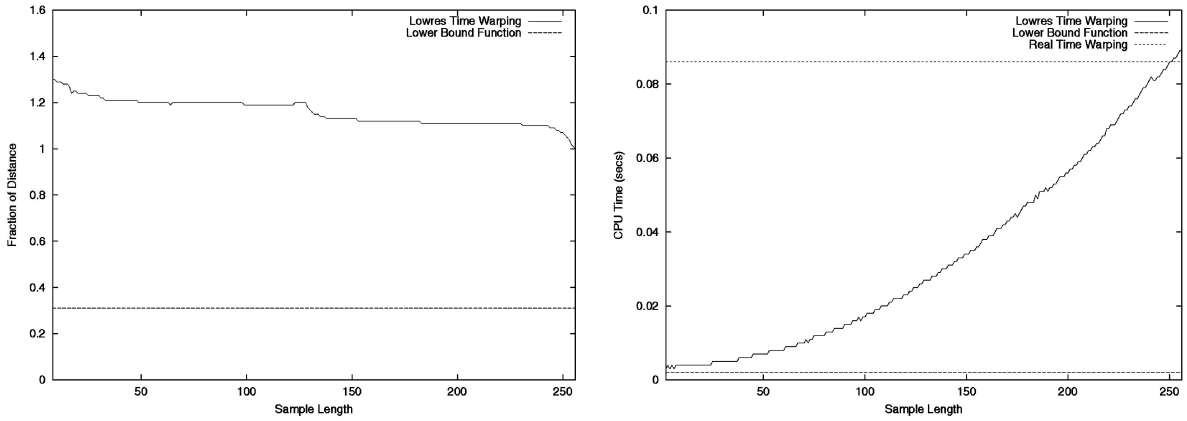
The approximation of low resolution to the time warping distance is found to be quite accurate. As observed, the fraction of estimation is close to 1.0 and is bounded between 1.0 and 1.3 for both data sets. Naturally, the larger the value of k_{sam} of low resolution time warping, the more accurate the approximation to the real-time warping distance.

The distance estimated by the lower bound distance function D_{lb} is shown for reference. As the lower bound distance function makes use of the full time series, the distance estimated is invariant to the sample length and is



(a) (b)

Fig. 9. (Real) (a) Fraction estimated. (b) CPU time versus sample length (sequence $\text{length} = 256$).



(a) (b)

Fig. 10. (Synthetic) (a) Fraction estimated. (b) CPU time versus sample length (sequence $\text{length} = 256$).

found to be around 0.65 (0.3), that is, 65 percent (30 percent) of the real-time warping distance for real (synthetic) data. Notice the performance gap of the lower bound distance function between the two data sets. This can be explained by examining some sample sequences as shown in Fig. 11. Both times series are extracted from the real stock data set. $\vec{x}$ is a typical sequence from the stock market. On the other hand, the shape of $\vec{y}$ seems to be constant, though they are from the same data source. The performance of D_{lb} is enhanced if the two sequences share the least overlapping range in the vertical axis [32]. We discovered that in the Hong Kong Stock Market, time series of type $\vec{y}$ is quite numerous. Therefore, D_{lb} performs better for this data set than for the synthetic data. For other stock markets, we expect the performance of the lower bound distance function would be similar for the real and synthetic data sets. Constant-valued sequences also have positive effect on the performance of low resolution time warping. Since Haar wavelets are highly effective in approximating constant sequence, less error is introduced in the resolution reduction process. Moreover, the approximated time values used in distance compensation will be close enough to the original ones. From Fig. 9a, we observe that $D_{lowresTW}$ of the real data set gets closer to the real-time warping distance than that of synthetic data. We observe that $D_{lowresTW}$ has a much better accuracy and more stable performance compared with D_{lb}.

The CPU time required by the above methods to match two sequences is shown in Figs. 9b and 10b. Both data sets share the same CPU time for series matching, since the computation time for the three methods relates to the length of the sequences only. The time warping distance computation consumes much more CPU time compared with the other two methods due to its $O(n^2)$ complexity. The CPU time of low resolution time warping increases with k_{sam}. For small values of k_{sam}, the CPU time is close to that of lower bound distance function. It gradually increases until it reaches the same CPU time required by real-time warping. Though the lower bound distance function achieves a lower CPU time, the distance estimated is not as effective as low resolution time warping for filtering purpose (0.3 - 0.65 versus 1.1 - 1.3). Moreover, it does not allow the trading off of CPU time for accuracy.

As CPU time of low resolution time warping increases with k_{sam}, we want to find a k_{sam} value that leads to fast computation time, while the distance estimation is close enough to real-time warping. A heuristic for determining the sample value would be k_{sam} values which offer the least CPU time increase for increased accuracy, we pick some portion with the slope close to zero in Figs. 9b and 10b. For sequence length of 256, the candidate k_{sam} values range from 3 to 25. A k_{sam} value of 25 would be preferable as it gives a closer estimation to real-time warping distance, though it shares almost the same CPU time with $k_{sam} = 3$. From simple observation and experimental evaluations as in Figs. 9b and 10b for other sequence lengths (lengths 64, 128, 512), a heuristics for the choice of k_{sam} for length n sequences can be generalized as below

$$k_{samH} = \lfloor \log_2 n \times 4 \rfloor, \tag{11}$$

where k_{samH} denotes the sample value obtained by the heuristic. Equation (11) is good enough to provide us a simple way to determine a sample value for fast computation of low resolution time warping while keeping a close

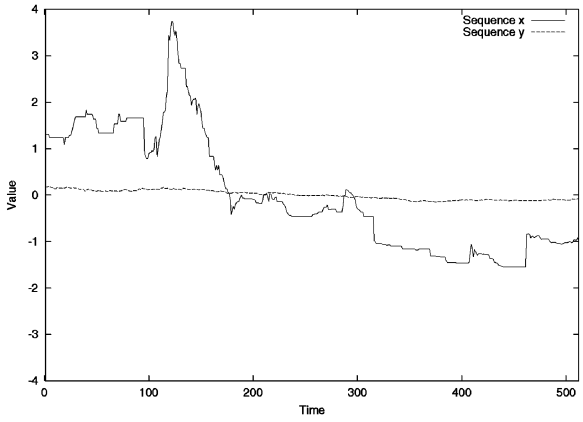
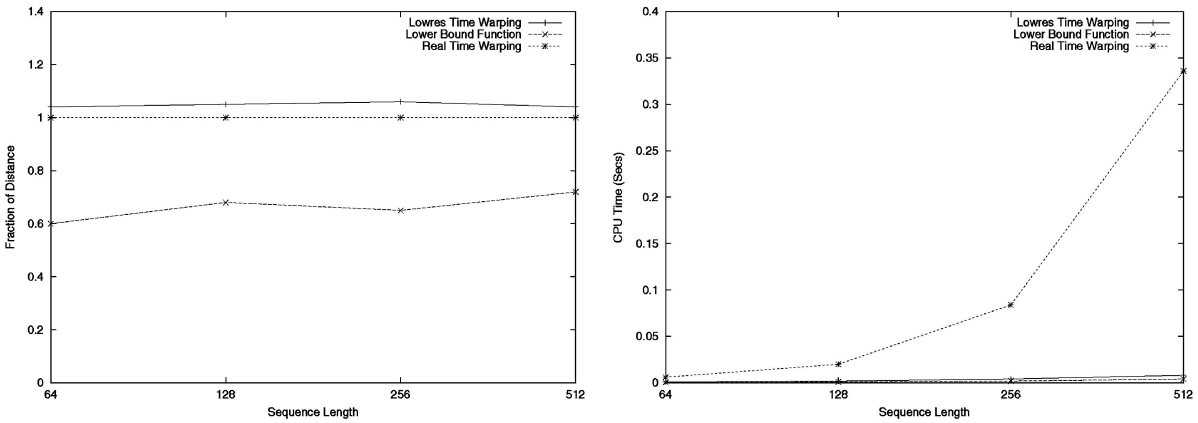


Fig. 11. Sample sequences from real stock database.



(a) (b)

Fig. 12. (Real) (a) Fraction estimated. (b) CPU time versus sequence length.

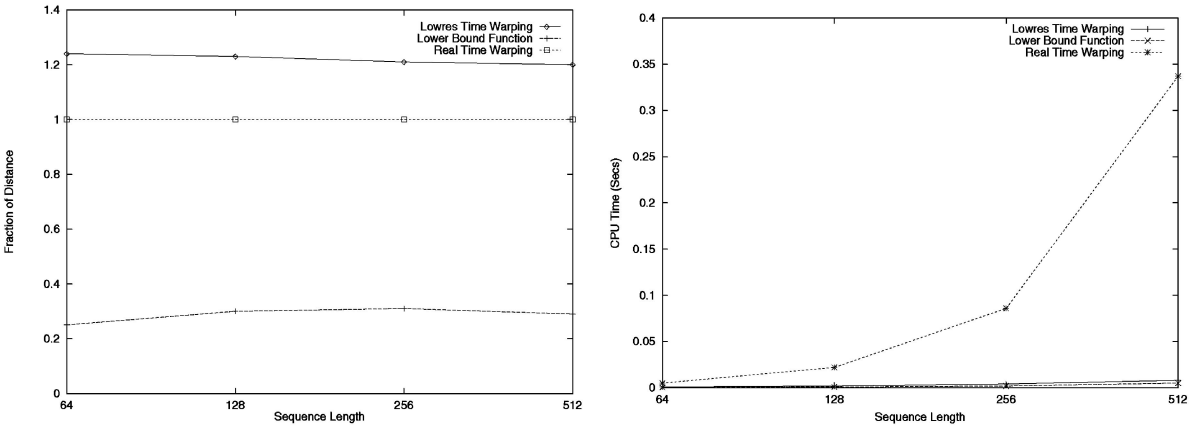
estimation to real-time warping distance. Unless specified otherwise, we use the k_{samH} as the sample values for all different lengths of sequences during the course of experimentation.

Figs. 12 and 13 show the fraction of distance estimated and the CPU time required for various sequence lengths. In Figs. 12a and13a, the fraction of distance that low resolution time warping estimated is relatively constant around the values 1.05 (1.20) for real (synthetic) data, while the estimation by lower bound distance function is close to 0.65 (0.3) for real (synthetic) data. In the figures, it is obvious that low resolution time warping is able to give a better approximation to real-time warping distance (an $+0.05/+0.20$ overestimation) compared with the lower bound distance function (an $-0.35/-0.70$ underestimation), and this phenomenon seems to persist in a variety of sequence lengths. This persistence is important in the sense that different lengths of sequences share nearly the same factor, though we are using different data sets for our experiments. Thus other sequence lengths will produce likely the same amount of over-estimation, which is used in turn to estimate the real-time warping distance.

Figs. 12b and 13b show the CPU time required for various sequence lengths. Having quadratic complexity, real-time warping scales badly in CPU time with sequence length. Meanwhile, both low resolution time warping and lower bound function maintain approximately a linear increase in CPU time and the advantage could be very significant for lengthy sequences. Both data sets consume about the same amount of CPU time for different lengths of sequences.

6.2 Precision versus Recall

The following experiments are carried out to evaluate the performance of low resolution time warping when acting as a filtering function in the postprocessing step of similarity search. We emphasize on the performance comparison of the filtering functions, hence we bypass the use of Fastmap index for the sake of simplicity. Performance is described in terms of precision and recall defined in (6). The number of false alarms as well as the number of false dismissals generated are also studied. The experimental setup is the same as before with a few changes. We perform a range search to look for candidate sequences. An appropriate



(a) (b)

Fig. 13. (Synthetic) (a) Fraction estimated. (b) CPU time versus sequence length.

TABLE 3
Fraction of Distance Estimated and Associated Standard Deviation

Seq. length n	Real		Synthetic		k_{samH}
	$dist\_frac$	S.D.	$dist\_frac$	S.D.	
64	1.04	0.09	1.25	0.11	24
128	1.05	0.10	1.24	0.13	28
256	1.06	0.09	1.21	0.10	32
512	1.04	0.06	1.21	0.12	36

search range $\epsilon_{timewarp}$ should be employed to obtain a reasonable amount of candidate sequences. In our experiment, the values of epsilon are set by choosing result sizes that range from 0.5 percent to 5 percent of the database size, and results are drawn from the average of results from these epsilon ranges.

The epsilon range $\epsilon_{timewarp}$ is then adjusted in accordance with the fraction of distance obtained in previous experiments (Figs. 12a and 13a). The adjusted epsilon is denoted as $\epsilon_{lowresTW}$. More false alarms but fewer false dismissals would appear for an increase in epsilon range. Our task is to obtain a range that results in a small number false dismissal, and at the same time suppresses the number of false alarms.

The fraction of distances estimated for various sequence lengths are shown in Figs. 12a and 13a, and tabulated in Table 3, accompanied with their standard deviations. The value of the fraction of distance estimated and its associated standard deviation act effectively as an indicator for the adjustment of the epsilon range $\epsilon_{lowresTW}$ in low resolution time warping. The adjustment can be expressed by the following equation:

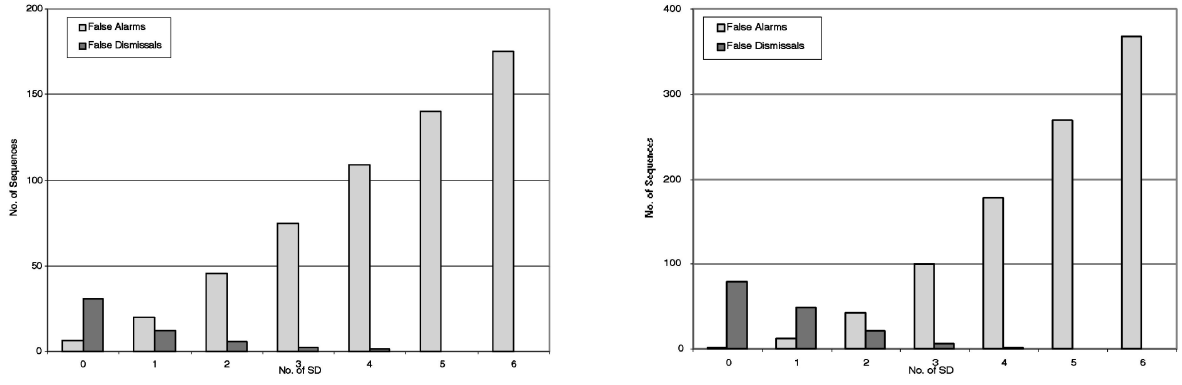
$$\epsilon_{lowresTW} = dist\_frac \times \epsilon_{timewarp} + c \times s.d. \quad \text{for } c \geq 0, \quad (12)$$

where $\epsilon_{timewarp}$ is the epsilon range used in real-time warping distance, $dist\_frac$ represents the fraction of distance estimated, and $s.d.$ is the standard deviation at a particular $dist\_frac$. By varying the value of c, the range $\epsilon_{lowresTW}$ can be systematically adjusted. The k_{samH} used in the experiments for low resolution time warping are listed in the same table for reference.

Figs. 14, 15, and 16 show the performance with various $\epsilon_{lowresTW}$. In Fig. 14, the number of false alarms and false dismissals are shown for querying a database of sequence length of 256. We observe that when we scale the value of the epsilon by a factor of $dist\_frac$ with zero $s.d.$, i.e., for real data, $\epsilon_{lowresTW} = 1.06 \times \epsilon_{timewarp}$, there are 34 false dismissals; for synthetic data, $\epsilon_{lowresTW} = 1.21 \times \epsilon_{timewarp}$, there are 85 false dismissals The performance of low resolution time warping for real data is better since $\epsilon_{lowresTW}$ for real data set is more closer to 1.0. The number of false alarms being generated is small. For $c = 6$, no false dismissal is recorded while the number of false alarms increases to 175 sequences for real data and 368 for synthetic data.

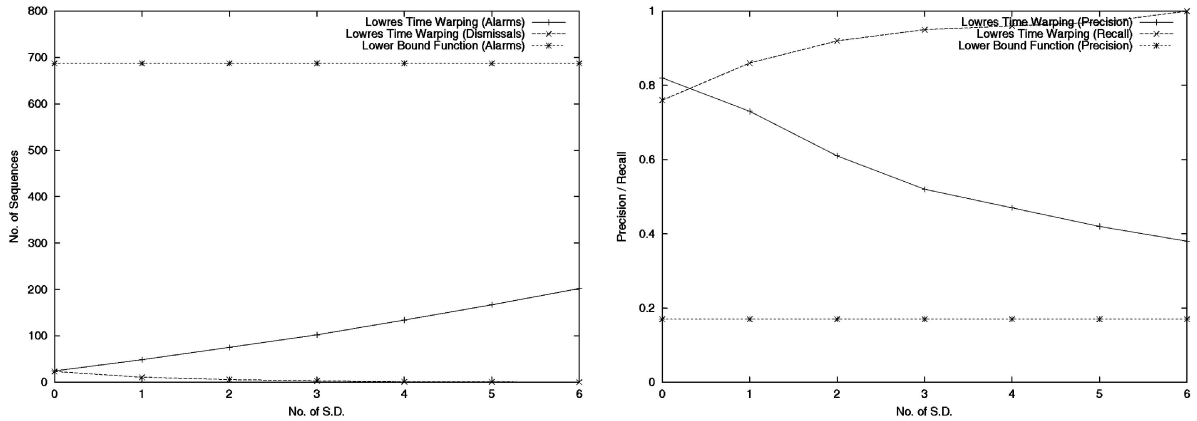
The same experiment is carried out separately for databases of sequence length of 64, 128, and 512. A similar trend is observed as in Fig. 14. Moreover, we notice that for the same value of c, there are more false alarms in database of short sequences. This phenomenon is explained later with Figs. 17a and 18a.

The fraction of real-time warping distance estimated by the lower bound distance function is not effective, although no false dismissal will be generated. This would lead to a large amounts of false alarms produced by D_{lb}, as demonstrated by experiment and shown in Figs. 15a and 16a. There are around 689 (3,000) false alarms being generated by D_{lb} for real (synthetic) data, compared with around 201 (503) for the low resolution method when recall $= 1.0$, we have achieved a 3.4 to six times improvement. The amount of false alarms produced by low resolution time



(a) (b)

Fig. 14. False alarms and dismissals versus synthetic data (sequence $\text{length} = 256$). (a) Real and (b) synthetic.



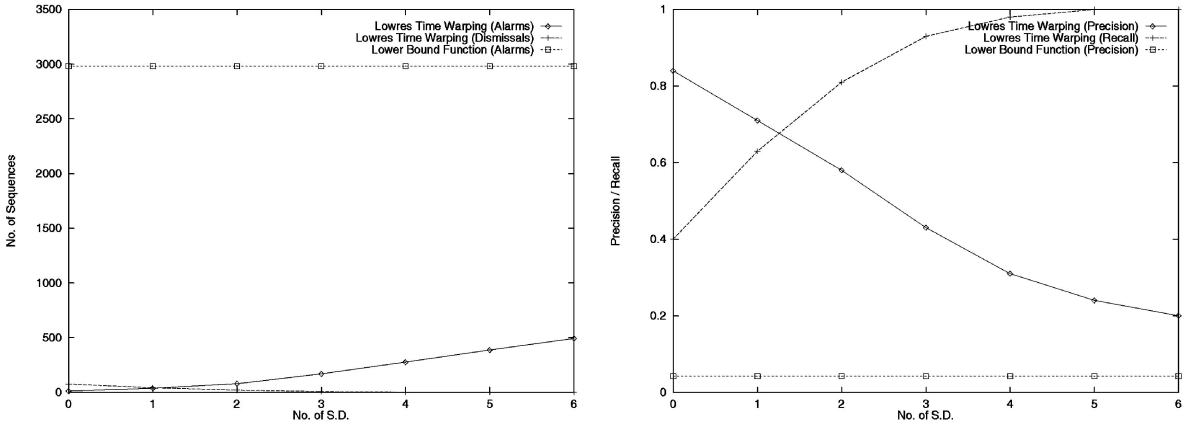
(a) (b)

Fig. 15. (Real) (a) False alarms. (b) Precision and recall versus synthetic data.

warping is still tiny for large standard deviation value with respect to D_{lb}. This leads to a significant performance gain by low resolution time warping technique, since most computations are involved in the matching between the candidate sequences and the query in the postprocessing step after the filtering. Having $O(n^2)$ complexity, the matching in real-time warping distance will consume a large amount of CPU time if the filtering function fails to prune away false alarms effectively.

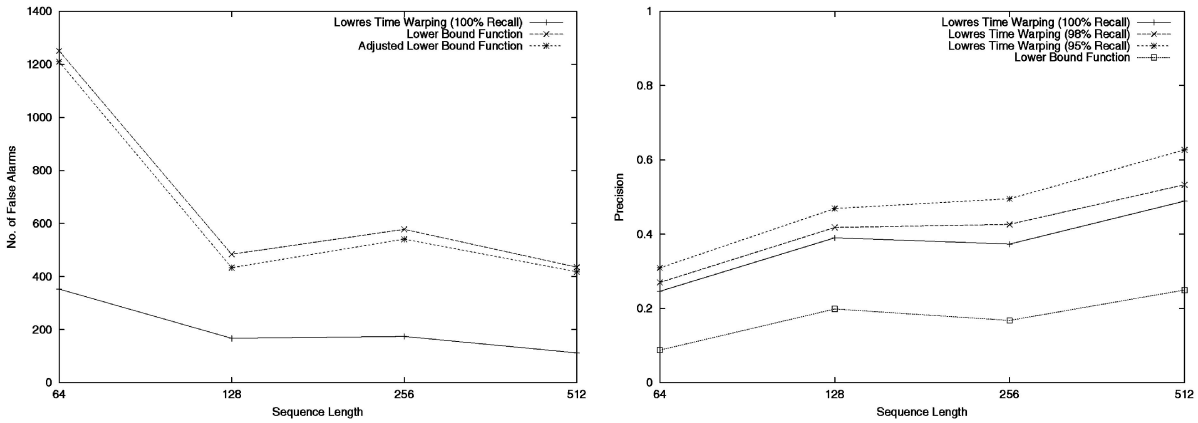
In Figs. 15b and 16b, the precision and recall of low resolution time warping are shown. The precision of lower bound function is also included for reference. For low resolution time warping, precision decreases while recall increases with $\epsilon_{lowresTW}$. Their trends correspond to the rise of false alarms and the drop of false dismissals, respectively, in Fig. 14. The value of recall recorded is 0.76 (0.40) for real (synthetic) data at $c = 0$, and gradually increases to 1.0 at $c = 6$, where no false dismissal occurs. Meanwhile, precision drops to 0.39 (0.21) for real (synthetic) data. Lower bound function has a much lower precision (0.18 for real data and 0.04 for synthetic data).

The performance of the two techniques are compared for a variety of sequence lengths in Figs. 17 and 18. For Figs. 17a and 18a, we show the number of false alarms generated by both filtering functions (recall $= 1.0$ for low resolution time warping). Both methods experience a decline in the number of false alarms with lengthy sequences. We observe that for lengthy sequences, the alignments of those peaks and valleys are rather localized, since the time series of financial data or random walk consist of time values fluctuating around some levels, which are relatively steady locally without abrupt change. Therefore, it is very unlikely that the head of a sequence will align with its tail. Thus, more nonqualified sequences can be filtered without appearing as false alarms. The performance gap is maintained for different sequence lengths. Compared to low resolution time warping, the lower bound function records 3.8 (12) times more false alarms for real (synthetic) data for sequence length of 512; and at least 3.4 (6.8) times



(a) (b)

Fig. 16. (Synthetic) (a) False alarms. (b) Precision and recall versus Synthetic data.



(a) (b)

Fig. 17. (Real) (a) False alarms. (b) Precision versus Sequence length.

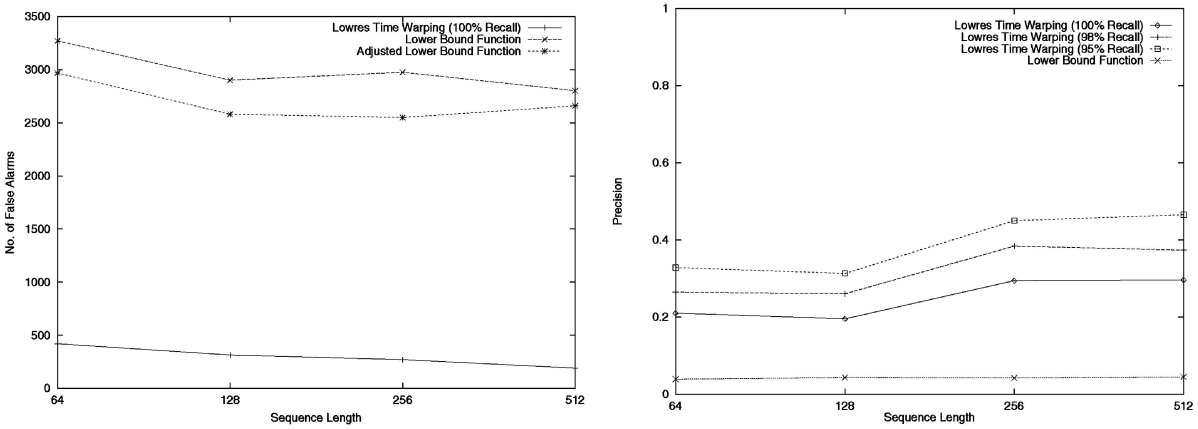
improvement for real (synthetic) data is recorded for sequence length of 64.

For fairness, we carry out experiments to discover any possible improvement when we adjust the epsilon of lower bound distance function, ϵ_{lb} based on (12). The results are also shown in Figs. 17a and 18a. The results are labeled *Adjusted Lower Bound Function*. Since the average lower bound distance function is much smaller than the real distance, we adjust the value of ϵ_{lb} to be the smallest value that can give a recall value of 1.0. For this setting, around 10 percent improvement is observed in the number of false alarms in both data sets. One may wonder why the number of false alarms remains large. The reason is that lower bound distance function does not try to approximate the time warping distance, but serves as a lower bound function. The average fraction estimated by D_{lb} is 0.65 (0.30) for real (synthetic) data (Figs. 9a and 10a), but the standard deviation is large so that ϵ_{lb} can only be reduced by a small amount. On the contrary, the standard deviation of lower resolution time warping is comparatively small and hence $\epsilon_{lowresTW}$ is highly effective in filtering.

The precision at various sequence lengths is depicted in Figs. 17b and 18b. While lower bound function maintains a low precision around 0.18 (0.04) for real (synthetic) data for different sequence lengths, low resolution time warping offers great improvements in precision, ranging from 0.25 to 0.6 in real data (0.2 to 0.45 for synthetic data) at recall $= 1.0$. The rise in precision for longer sequences directly corresponds to the drop in the number of false alarms in Figs. 17a and 18a. In addition, we show the precision of low resolution time warping at recall $= 0.98$ and 0.95. Smaller recall value provides better precision, since the epsilon range $\epsilon_{lowresTW}$ is reduced.

6.3 Overall Runtime

In this experiment, we measure the overall CPU time of similarity search based on our strategy. The time measurement commences when the query is posed and ends when all result sequences are returned from the postprocessing step. We do not include the steps of K-L transform and index search as described in Section 4.1 since these steps, if applied, will be common to the lower bound method and



(a) (b)

Fig. 18. (Synthetic) (a) False alarms. (b) Precision versus Sequence length.

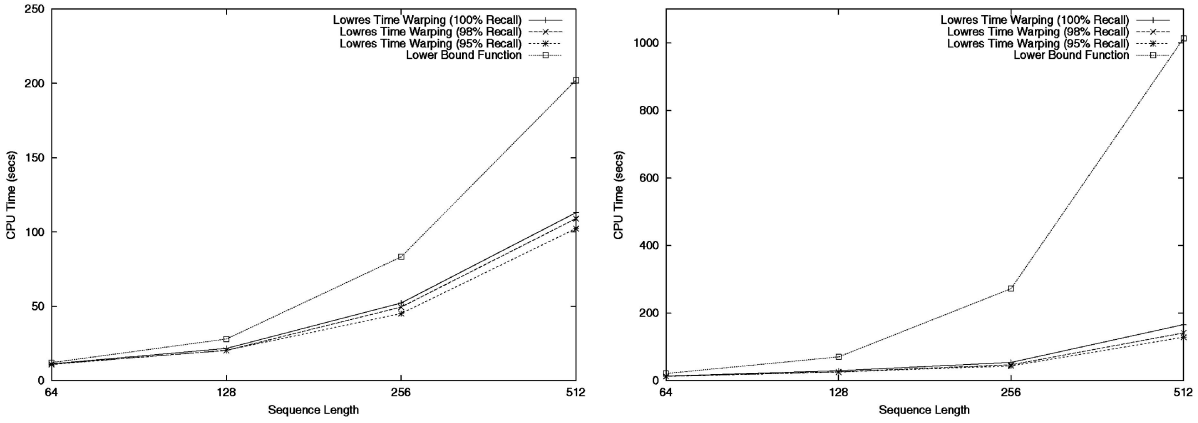


Fig. 19. Overall CPU time versus Sequence length (a) Real and (b) Synthetic.

the low resolution method. Experimental setup is kept the same as in Section 6.2. Results are shown in Fig. 19.

The CPU time for different time series lengths is shown in Fig. 19. Low resolution time warping outperforms lower bound distance function significantly, especially for lengthy sequences. It has a much better scaling with sequence length. Numerically, a 1.9 to 5.4 times improvement is achieved for sequence length of 512. The trend suggests the improvement would increase for lengthy sequences. As expected, low recall value of low resolution time warping consumes less overall CPU time at different lengths of sequences. In addition, we examine both data sets with the sequential search method. The CPU time of sequential search recorded at sequence length of 512 is 1,684 (1,690) secs for real (synthetic) data, which represents a 15 (10.3) times improvement.

6.4 Starting Up Evaluation

Note that in our performance evaluation, the two important indicators, namely the fraction of distance estimated in Figs. 9 and 10, and the best ϵ range $\epsilon_{lowresTW}$ in (12) are obtained *statistically*. These results are valid for our particular time series database. However, for other sets of time series data, results may vary accordingly. Therefore, before starting up a new similarity querying database based on our strategy, the two sets of experimental analysis should be carried out as in our evaluation to find the values of these indicators. In case of very large database, sampling of sequences could be adopted to reduce the running time of the evaluation process.

7 Conclusion

As time series data are of growing importance, we need to support similarity search in sequence data in database systems. We make a study of two different ways of measuring similarity: Euclidean distance and time warping distance. For Euclidean distance measurement, traditional method reduces the dimensionality by means of DFT. We propose to use the Haar wavelet transform instead and show that it can preserve Euclidean distance provided that we apply a proper normalization factor. Therefore, this technique can guarantee no false dismissal during the search. Experiments show that this method has a much more efficient preprocessing step and has competitive performance in the accuracy.

While Euclidean distance is frequently adopted, another extensively used time warping distance can handle the time shifts problems in similarity matching of financial series. Two major problems of this technique are the high complexity and the fact that it is not a metric distance. False dismissals are produced when we apply metric indexing techniques. A previous technique applies such an index anyways followed by a postprocessing step to extract the results. Since time warping distance is computationally expensive, a filtering function is first applied in the postprocessing step to filter some false alarms. We follow the above basic approach but propose a different filtering function based on Haar wavelet transformation. We suggest using a low resolution approximation of the real-time warping distance which is easy to compute but closely approximates the time warping distance. Experiments show that the proposed method.

Acknowledgments

The authors would like to thank the anonymous referees for their careful review and very constructive suggestions and comments on the initial draft of the paper. This research was supported by the RGC (the Hong Kong Research Grants Council) grant UGC REF.CUHK 4436/99E.

References

[1] J.I. Agbinya, "Discrete Wavelet Transform Techniques in Speech Processing," *Proc. IEEE TENCON—Digital Signal Processing Applications Conf.*, pp. 514-519, 1996.
[2] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," *Proc. Fourth Int'l Conf. Foundations of Data Organization and Algorithms,*" 1993.
[3] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. Int'l Conf. Data Eng.*, pp. 3-14, Mar. 1995.
[4] A.N. Akansu and R.A. Haddad, *Multiresolution Signal Decomposition.* Academic Press, 1992.
[5] J.J. Benedetto and M.W. Frazier, *Wavelets—Math. and Applications.* CRC, 1994.

[6] D.J. Berndt and J. Clifford, *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1995.
[7] C.S. Burrus, R.A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms, A Primer*. Prentice Hall, 1997.
[8] K.P. Chan and A. Fu, "Efficient Time Series Matching by Wavelets," *Proc. Int'l Conf. Data Eng.*, 1999.
[9] C. Chatfield, *The Analysis of Time Series: An Introduction*. Chapman and Hall, 1984.
[10] T. Edwards, "Discrete Wavelet Transforms: Theory and Implementation," technical report, Stanford Univ., 1991.
[11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases," *Proc. ACM SIGMOD Conf. Management of Data*, pp. 419-429, 1994.
[12] J. Foran, *Fundamentals of Real Analysis*. Marcel Dekker, 1991.
[13] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*. Addison Wesley, 1992.
[14] K. Grochenig and W. R. Madych, "Multiresolution Analysis, Haar Bases, and Self-Similar Tilings of r^n," *IEEE Trans. Information Theory*, vol. 38, no. 2, pp. 556-568, 1992.
[15] K.V.R. Kanth, D. Agrawal, and A. Singh, "Dimensionality Reduction for Similarity Searching in Dynamic Databases," *Proc. ACM SIGMOD Conf. Management of Data*, 1998.
[16] E.J. Keogh and M.J. Pazzani, "Scaling Up Dynamic Time Warping for Datamining Applications," *Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, 2000.
[17] F. Korn, H.V. Jagadish, and C. Faloutsos, "Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences," *Proc. ACM SIGMOD Conf. Management of Data*, 1997.
[18] C.-S. Li, P.S. Yu, and V. Castelli, "Hierarchyscan: A Hierarchical Similarity Search Algorithm for Databases of Long Sequences," *Proc. Int'l Conf. Data Eng.*, 1996.
[19] C.S. Myers and L.R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, 1981.
[20] A. Natsev, R. Rastogi, and K. Shim, "Walrus: A Similarity Retrieval Algorithm for Image Databases," *Proc. ACM SIGMOD Conf. Management of Data*, pp. 395-406, 1999.
[21] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*. Prentice Hall, 1975.
[22] P. Sanghyun, W. Chu, J. Yoon, and C. Hsu, "Efficient Similarity Searches for Time-Warped Subsequences in Sequence Databases," *Proc. Int'l Conf. Data Eng.*, 2000.
[23] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice Hall, 1993.
[24] D. Rafiei and A. Mendelzon, "Similarity-Based Queries for Time Series Data," *Proc. ACM SIGMOD Conf. Management of Data*, pp. 13-25, 1997.
[25] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," *Proc. ACM SIGMOD Conf. Management of Data*, pp. 71-79, 1995.
[26] H. Sakoe, "Two-Level DP-Matching—A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 27, no. 6, 1979.
[27] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *Readings in Speech Recognition*, Morgan Kaufmann Publishers, 1990.
[28] D. Shasha, *Time Series in Finance: The Array Database Approach*. http://www.cs.nyu.edu/cs/faculty/shasha/papers/jagtalk.html. Apr. 2000.
[29] E.J. Stollnitz, T.D. Derose, and D.H. Salesin, *Wavelets for Computer Graphics*. Morgan Kaufmann, 1996.
[30] D. Wu, D. Agrawal, A. El Abbadi, A. Singh, and T.R. Smith, "Efficient Retrieval for Browsing Large Image Databases," *Proc. Conf. Information and Knowledge Management*, 1996.
[31] B.-K. Yi and C. Faloutsos, "Fast Time Sequence Indexing for Arbitrary L_p Norms," *Proc. 26st VLDB Conf.*, pp. 385-394, 2000.
[32] B.-K. Yi, H.V. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences under Time Warping," *Proc. Int'l Conf. Data Eng.*, 1998.

Franky Kin-Pong Chan received the BE degree in system engineering in 1997 and the MPhil degree in computer science and engineering, in 1999, both from the Chinese University of Hong Kong. Currently, he is a research engineer at the University of Hong Kong. His main research interests are in similarity search for time series, multidimensional index, high-performance security on mobile platform, multimedia content distribution, and digital right management.

Ada Wai-chee Fu received the BSc degree in computer science from the Chinese University of Hong Kong in 1983, and the Msc and PhD degrees in computer science from Simon Fraser University of Canada in 1985 and 1990, respectively. She has been a member of the scientific staff at Bell Northern Research of Canada from 1989 to 1993. Since 1993, she has been a faculty member of the Department of Computer Science and Engineering at the Chinese University of Hong Kong. Her interests include topics in database systems and parallel and distributed systems. She is a member of the IEEE and the ACM.

Clement Yu received the BS degree in applied mathematics from Columbia University and the PhD degree in computer science from Cornell University. He is a professor of computer science at the University of Illinois at Chicago. He serves/served as an associate editor/member of editorial board of *IEEE Transactions On Knowledge and Data Engineering*, *Journal of Distributed and Parallel Databases*, *International Journal of Software Engineering and Knowledge Engineering*, *WWW Internet and Web Information System*, and *IEEE Transactions on Multimedia*. He also served as chair of the ACM Special Interest Group on Information Retrieval, program chair/general chair of a number of conferences/workshops, and as an advisory committee member of the US National Science Foundation. He is a co-author of the book *Principles of Database Query Processing for Advanced Applications* which was published by Morgan Kaufmann. His research interests are information (multimedia) retrieval and database management.