Efficient Time Series Matching by Wavelets

Kin-pong Chan and Ada Wai-chee Fu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
{kpchan, adafu}@cse.cuhk.edu.hk

Abstract

Time series stored as feature vectors can be indexed by multi-dimensional index trees like R-Trees for fast retrieval. Due to the dimensionality curse problem, transformations are applied to time series to reduce the number of dimensions of the feature vectors. Different transformations like Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), Karhunen-Loeve (K-L) transform or Singular Value Decomposition (SVD) can be applied. While the use of DFT and K-L transform or SVD have been studied in the literature, to our knowledge, there is no in-depth study on the application of DWT. In this paper, we propose to use Haar Wavelet Transform for time series indexing. The major contributions are: (1) we show that Euclidean distance is preserved in the Haar transformed domain and no false dismissal will occur, (2) we show that Haar transform can outperform DFT through experiments, (3) a new similarity model is suggested to accommodate vertical shift of time series, and (4) a two-phase method is proposed for efficient n-nearest neighbor query in time series databases.

1. Introduction

Time series data are of growing importance in many new database applications, such as data warehousing and data mining [3, 8, 2, 12]. A time series (or time sequence) is a sequence of real numbers, each number representing a value at a time point. Typical examples include stock prices or currency exchange rates, biomedical measurements, weather data, etc ... collected over time. Therefore, time series databases supporting fast retrieval of time series data and similarity queries are desired.

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

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

By using this similarity model, we can retrieve similar time series by considering distance $D(\vec{x}, \vec{y})$.

Indexing is used to support efficient retrieval and matching of time series. Some important factors have to be considered: The first factor is *dimensionality reduction*. Many multi-dimensional indexing methods [13, 7, 5, 20] such as the R-Tree and R*-Tree [20, 5, 11] scale exponentially for high dimensionalities, eventually reducing the performance to that of sequential scanning or worse. Hence, transformation is applied to map the time sequences to a new feature space of a lower dimensionality. Next we must ensure *completeness and effectiveness* when the number of dimensions is reduced. To avoid missing any qualifying object, the Euclidean distance in the reduced k-dimensional space should be *less than* or *equal* to the Euclidean distance between the two original time sequences. Finally, we must also consider the *nature of data series* since the effectiveness of *power concentration* of a particular transformation depends on the nature of the time series. 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 [10], for which a skewed energy spectrum can be obtained.

Discrete Fourier Transform (DFT) has been one of the most commonly used techniques. One problem with DFT is that it misses the important feature of time localization. *Piecewise Fourier Transform* has been proposed to mitigate this problem, but the size of the *pieces* leads to other problems. While large *pieces* reduce the power of multi-resolution, small *pieces* has weakness in modeling low frequencies.

Wavelet Transform (WT), or *Discrete Wavelet Transform* (DWT) [9, 18] has been found to be effective in replacing DFT in many applications in computer graphics, image [26], speech [1] , and signal processing [6, 4]. We propose to apply this technique in time series for dimension reduction and content-based search. DWT is a discrete version of WT for numerical signal. Although the potential application of DWT in this problem was pointed out in [22], no further investigation has been reported to our knowledge. Hence, 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 *multi-resolution* representation of signals. It has the *time-frequency localization* property. Thus, DWT is able to give locations in both time and frequency. Therefore, wavelet representations of signals bear more information than that of DFT, in which only frequencies are considered. While DFT extracts the lower harmonics which represent the general shape of a time sequence, DWT encodes a coarser resolution of the original time sequence with its preceding coefficients. We show that Euclidean distance is preserved in the Haar transformed domain. Moreover, we show by experiments that *Haar Wavelet Transform* [1] [9], which is a commonly used wavelet transform, can outperform DFT significantly.

We also suggest a similarity definition to handle the problem of vertical shifts of time series. Finally we propose an algorithm on n-nearest neighbor query for the proposed wavelet method. The algorithm makes use of the range query and dynamically adjusts the *range* by the property of Euclidean distance preservation of the wavelet transformation.

2. Related Work

Discrete Fourier Transform (DFT) is often used for dimension reduction [2, 15] to achieve efficient indexing. 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 $\vec{X}$. For many applications such as stock data, the low frequency components are located at the preceding coefficients of $\vec{X}$ which represent the general trend of the time sequence $\vec{x}$. These coefficients can be indexed in an R-Tree or R*-Tree for fast retrieval. 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.

Parseval's Theorem [23] shows that the Euclidean distance between two signals $\vec{x}$ and $\vec{y}$ in time domain is the same as their Euclidean distance in frequency domain

$$\|\vec{x} - \vec{y}\|^2 \equiv \|\vec{X} - \vec{Y}\|^2 \tag{1}$$

Therefore, F-index may raise false alarms, but guarantees no false dismissal. After a range query in the F-index, false alarms are filtered by checking against the query sequence in the original time domain in a post-processing step. F-index is further generalized and subsequence matching is proposed in [15]. This is called the ST-index which permits sequence query of varying length. Each time sequence is broken up into pieces of subsequences by a sliding window with a fixed length ω for DFT. Feature points in nearby offsets will form a trail due to the effect of stepwise sliding window, the *minimum bounding rectangle* (MBR) of a trail is then being indexed in an R-Tree instead of the feature points themselves. When a query arrives, all MBRs that intersect the query region are retrieved and their trails are matched.

New similarity models are applied to F-index based time series matching in [24]. It achieves time warping, moving average, and reversing by applying transformations to feature points in the frequency domain. Given a query $\vec{q}$, a new index is built by applying a transformation to all points in the original index and feature points with a distance less than ϵ from $\vec{q}$ are returned. However, a lot of computations are involved in building the new index. which has a great impact on the actual query performance.

In the above works, no efficient method for nearest neighbor query, which can be more useful than range query, has been proposed.

[1] We shall use Haar wavelet transform and DWT interchangeably throughout this paper, unless specified particularly.

Another method that has been employed for dimension reduction is *Karhunen-Loeve* (K-L) transform [28]. (This method is also known as Singular Value Decomposition (SVD) [22], and is called Principle Component analysis in statistical literature.) Given a collection of n-dimensional points, we project them on a k-dimensional sub-space 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. Therefore, new projection matrix should be re-calculated and the index tree has to be re-organized periodically to keep up the search performance.

2.1. Wavelet Transform

Wavelets are basis functions used in representing data or other functions. Wavelet algorithms process data at different *scales* or *resolutions* in contrast with DFT where only frequency components are considered. The origin of wavelets can be traced to the work of Karl Weierstrass [27] in 1873. The construction of the first orthonormal system by Haar [21] is an important milestone. Haar basis is still a foundation of modern wavelet theory. Another significant advance is the introduction of a nonorthogonal basis by Dennis Gabor in 1946 [16]. In this work we shall advocate the use of the Haar wavelets in the problem of time series retrieval.

3. The Proposed Approach

Following a trend in the disciplines of signal and image processing, we propose to study the use of wavelet transformation for the time series indexing problem. Before we go into the details of our proposed techniques, 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 time sequences $\vec{x}$ and $\vec{y}$ of equal length n are said to be* **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 Figure 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 time sequences $\vec{x}$ and $\vec{y}$ of equal length n are said to be* **v-shift similar** *if*

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

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

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.

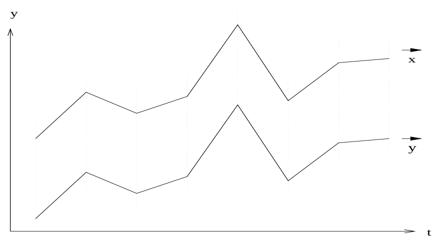


Figure 1. Example of vertical shifts of time sequences

3.1. Haar Wavelets

We want to have a decomposition that is fast to compute and requires little storage for each sequence. The Haar wavelet is chosen for the following reasons: (1) it allows good approximation with a subset of coefficients, (2) it can be computed quickly and easily, requiring linear time in the length of the sequence and simple coding, and (3) it preserves Euclidean distance (see Section 3.3). The formal definition of *Haar wavelets* is given in Appendix A. Concrete mathematical foundations can be found in [9, 19] and related implementations in [14].

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 $f(x)$. The procedure to find the Haar transform of a discrete function $f(x)$ = (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 $f(x)$. 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 $H(f(x))$ = (c d_0^0 d_0^1 d_1^1) = (6 2 1 -1) is obtained which is composed of the last average value 6 and the coefficients found on the right most column, 2, 1 and -1. It should be pointed out that c 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 difference values 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.

Haar transform can be realized by a series of matrix multiplications as illustrated in Equation (2). Envisioning the example input signal $\vec{x}$ as a column vector with length [2] n = 4, an intermediate transform vector $\vec{w}$ as another column vector and Haar transform matrix $\mathbf{H}$

$$\begin{bmatrix} x_0' \\ d_0^1 \\ x_1' \\ d_1^1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2}$$

The factor 1/2 associated with the Haar transform matrix can be varied according to different *normalization* [3] conditions. After the first multiplication of $\vec{x}$ and $\mathbf{H}$, half of the Haar transform coefficients can be found which are d_0^1 and d_1^1 in $\vec{w}$ interleaving with some intermediate coefficients x_0' and x_1'. Actually, d_0^1 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' x_1' 0 0]^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 this particular case, c and d_0^0 can be found in the second iteration.

The complexity of Haar transform can be evaluated by considering the number of operations involved in the recursion process.

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

Proof: There are totally n matrix additions *or* subtractions in the first iteration of matrix operation. The size of the input vector is halved in each iteration onwards. The total number of operations is formulated as

$$\overbrace{n + n/2 + \cdots + 2}^{\log_2 n} = 2\frac{2^{\log_2 n} - 1}{2-1} = 2(n-1)$$

which is bounded by $O(n)$. □

3.2. DFT versus Haar Transform

Our motivation of using Haar transform to replace DFT is based on several evidences and observations, some of which are also the reasons why the use of wavelet transforms instead of DFT is considered in areas of image and signal processing.

The first reason is on the pruning power. The nature of the Euclidean distance preserved by Haar transform and DFT are different. In DFT, comparison of two time sequences is based on their low frequency components, where most energy is presumed to be concentrated on. On the other hand, the comparison of Haar coefficients is matching a gradually refined resolution of the two time sequences. From intuition, Euclidean distance can be highly related to low resolution of signal rather than low frequency components. This property can give rise to more effective pruning, i.e. less false alarms will appear, which is confirmed by experiments in Section 5.

Another reason is the complexity consideration. The complexity of Haar transform is $O(n)$ whilst $O(n \log n)$ computation is

[2] As for Fast Fourier Transform, the length of the signal is restricted to numbers which are power of 2.

[3] The normalization is described in Section 3.3.

required for Fast Fourier Transform (FFT) [17]. Both impose restriction on the length of time sequences which must be an integral power of 2. Although these computations are all involved in pre-processing stage, the complexity of the transformation can be a concern especially when the database is large. From our experiments, the pre-processing time for DFT is about 3 to 4 times longer than Haar transform.

Finally, the proposed method provides better similarity model. Apart from Euclidean distance, our model can easily accommodate *v-shift* similarity of two time sequences (Definition 2) at a little more cost. That is, the situation where vertically shifted signals can match is accommodated. On the other hand, previous study on F-index did not make use of this similarity model.

Note that similar to DFT, DWT will not require massive index re-organization because of database updating, which is a major drawback in using the K-L transform or SVD approach.

3.3. Guarantee of no False Dismissal

For FT and DFT, it is shown by Parseval's Theorem [23] that the energy of a signal conserves in both time and frequency domains. Parseval's Theorem also shows that this situation is true for wavelet transforms. On the other hand, the Euclidean distances of both time and frequency domains are the same for DFT by Equation (1). 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. However, this property has not been shown for DWT in general, and not for the Haar wavelets. Here we show such a relationship.

Lemma 2 *Given a sequence $\vec{x}$ = $(x_0\ x_1)$ and a sequence $\vec{y}$ = $(y_0\ y_1)$. 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 $D(\vec{s},\vec{r})$, i.e. $D(\vec{x},\vec{y}) = \sqrt{2}D(\vec{s},\vec{r})$*

Proof: Express $\vec{s}$ in terms of $\vec{x}$ and $\vec{r}$ in terms of $\vec{y}$ by applying Equation (2) accordingly.

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

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

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

Thus, $D^2(\vec{s},\vec{r}) = (D^2(\vec{x},\vec{y}))/2$, and $D(\vec{x},\vec{y}) = \sqrt{2}D(\vec{s},\vec{r})$ □

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*

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

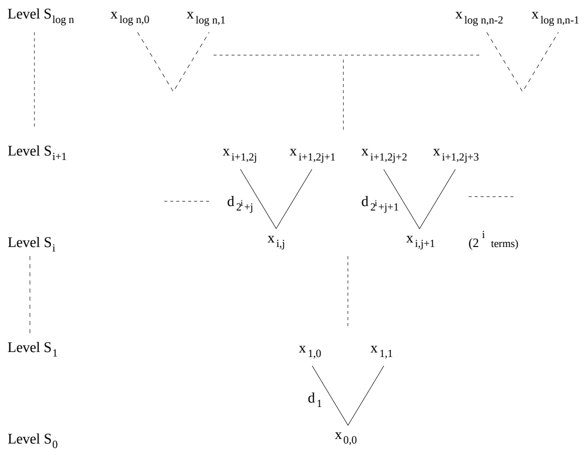


Figure 2. Hierarchy of Haar wavelet transform of sequence $\vec{x}$ of length n

Proof: In Figure 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

$$x_{i,j} = \frac{x_{i+1,2j} + x_{i+1,2j+1}}{2}$$

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

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 $D_i = d_i$ of sequence $\vec{x} - d_i$ 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 S_i contains data $\{x_{i,0}, x_{i,1}, \ldots, x_{i,2^i-1}\}$. Let us define S_i to be

$$S_i = \sqrt{\sum_{j=0}^{2^i-1} (x_{i,j} - y_{i,j})^2}$$

S_i can be seen as the Euclidean distance between the data sequences at level i ($0 \leq i \leq \log_2 n$) 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:

$$\begin{aligned} S_{i+1} &= \sqrt{2 \times (S_i^2 + D_{2^i}^2 + D_{2^i+1}^2 + \cdots + D_{2^{i+1}-1}^2)} \\ &\quad \text{for } 0 \leq i \leq \log_2 n - 1 \\ S_0 &= C \end{aligned} \qquad (4)$$

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

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

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 > 0 of the form $\{x_{i+1,2j}, x_{i+1,2j+1}\}$, we have the following relation

$$(x_{i+1,2j}-y_{i+1,2j})^2+(x_{i+1,2j+1}-y_{i+1,2j+1})^2 = 2\left((x_{i,j}-y_{i,j})^2+\left(d_{2^i+j}-d'_{2^i+j}\right)^2\right) \quad (5)$$

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 $i=k$,

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

By Equation (5), we have S_{k+1}

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

Finally by definition of S_k,

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

which completes the proof. □

The expression of the Euclidean distance between time sequences in terms of their Haar coefficients is not sufficient for proper use in multi-dimensional index trees until Euclidean distance preserves in both Haar and time domains, as for DFT in (1). This can be achieved by a normalization step which replaces the scaling factor in Equation (2) from $1/2$ to $1/\sqrt{2}$ in the Haar transformation. After the normalization step, Euclidean distance between sequences in Haar domain will be equivalent to $S_{\log_2 n}$ in Equation (3). The preservation of Euclidean distance of Haar transform ensures the completeness of feature extraction as in DFT.

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

Lemma 4 *If the first h_c ($1 \le h_c \le n$) dimensions of Haar transform are used, no false dismissal will occur for range queries.*

Proof: Considering the inequality in Definition 1 and Lemma 3

$$D(\vec{x},\vec{y}) = S_{\log_2 n} \le \epsilon \quad (6)$$

Using the first h_c dimensions as index, the value of D_i in Equation (3) will become zero for $i \ge h_c$. Thus the Euclidean distance between two sequences is $\le S_{\log_2 n} \le \epsilon$. This completes the proof. □

4. The Overall Strategy

In this section, we present the overall strategy of our time series matching method and propose our own method for nearest neighbor query. Before querying is performed, we shall do some pre-processing 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.

4.1. Pre-processing

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 to time series, but the first Haar coefficient will not be used in indexing, as there is no need to match their average values.

Step 2 - Index Construction: Given a database of time series of varying length. We pre-process the time series as follows. We obtain the ω-point Haar transform by applying Equation (2) with the normalization factor, for each subsequences with a sliding window of size ω to 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 post-processing cost and index dimension.

4.2. Range Query

After we have built the index, we can carry out range query or nearest neighbor query evaluation. For range queries, two steps are involved:

1. Similar sequences with distances $\le \epsilon$ from the query are looked up in the index and returned.
2. A post processing step is applied on these sequences to find the true distances in time domain to remove all false alarms.

4.3. Nearest Neighbor Query

For nearest neighbor query, we propose a two-phase evaluation as follows.

- *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) are computed between the query sequence and all n nearest neighbors obtained which are $D(\vec{q}, \vec{nn}_i^1)$, where $\vec{nn}_i^1$ denotes the nearest neighbor i ($1 \le i \le n$), 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,

[4] Using Definition 2, one dimension can be saved in the index tree.

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 post processing step mentioned in Section 4.2 is avoided since the Euclidean distances are found already in time domain during the search. In the search we keep updating [5] the value of ϵ with $D(\vec{q}, \vec{nn}_n^2)$ which is the distance of the current farthest neighbor among the n nearest neighbors. The n nearest neighbors stored in the list are returned as answer when the range query evaluation is finished. The distance of the farthest nearest neighbor with query $\vec{q}$ is $D(\vec{q}, \vec{nn}_n^{ans})$.

The correctness of the above algorithm can be shown by considering two cases. For the first case, assume the n nearest neighbors in the final answer all appear in the results in Phase 1, $\vec{nn}_i^1$ = $\vec{nn}_j^{ans}$, where $1 \leq i, j \leq n$ and i need not be equal to j. Obviously, $D(\vec{q}, \vec{nn}_n^1)$ = $D(\vec{q}, \vec{nn}_n^{ans})$. In the second case, assume some or no nearest neighbor obtained in the final answer appears in the results in Phase 1, $\vec{nn}_i^1 \neq \vec{nn}_j^{ans}$, where $1 \leq i, j \leq n$ for some i and j. Thus, $D(\vec{q}, \vec{nn}_n^1) > D(\vec{q}, \vec{nn}_n^{ans})$. Therefore, $D(\vec{q}, \vec{nn}_n^1) \geq D(\vec{q}, \vec{nn}_n^{ans})$ and by Lemma 4 there are only false alarms produced in the range query of Phase 2 since the value of ϵ upper bounds the distance of the farthest neighbor $\vec{nn}_n^{ans}$.

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 small query range to prune out a large amount of candidates for 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 non-qualifying MBRs during the traversal of R-Tree.

5. Performance Evaluation

Experiments using real stock data and synthetic random walk data have been carried out. All experiments are conducted on a Sun UltraSPARC-1 workstation with 686MBytes of main memory. Page size is set to 1024 bytes. A branching factor of 20 is chosen for the R-Tree so that the index tree nodes can be fitted within one disk page. We pointed out earlier that pre-processing time for Haar wavelet is much less than that for DFT. Here we shall compare the querying performance.

We have experimented with both real data and synthetic data. Real data are extracted from different equities of Hong Kong stock market from 12/7/90 to 7/11/96. They have been collected daily over the time period. Totally 10k feature vectors are extracted by a sliding window of size ω = 512 and inserted into an R-Tree.

Both range and nearest neighbor queries are examined and the results are shown in Figures 3 - 6. Random queries are applied with varying epsilon ϵ, which ranges from 0.5% to 5% of the database size. The number of nearest neighbors for nearest neighbor query is between 20 and 40. All results are obtained from the

[5] The updating process begins only when the list storing the nearest neighbors has been filled up already.

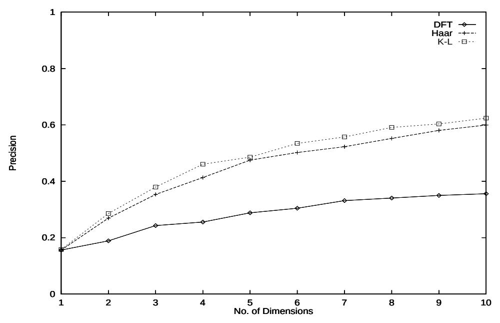


Figure 3. Precision of Range query

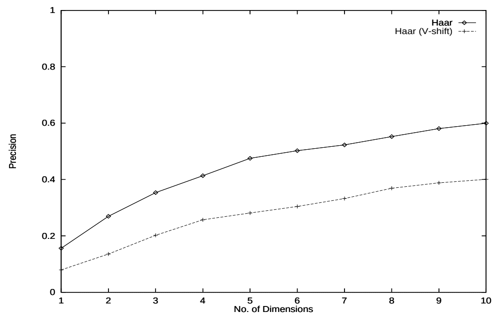


Figure 4. Precision of Range query (V-shift model)

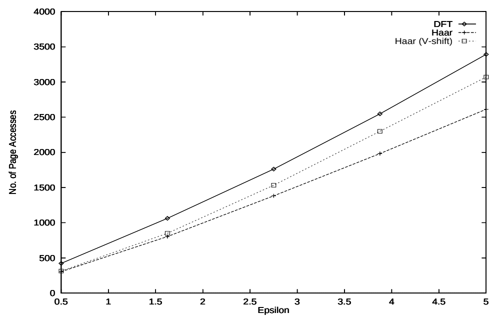


Figure 5. Page accesses of Range query

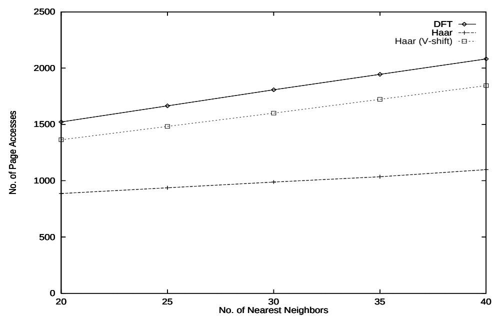


Figure 6. Page accesses of NN query

average of 100 trials. In each figure, Haar transforms using Definition 1 and Definition 2 as similarity models are denoted as 'Haar' and 'Haar(V-shift)' respectively.

In Figure 3, precision against the first tenth indexed coefficients/dimensions is investigated using Definition 1. It is defined as

$$\text{Precision} = \frac{S_{time}}{S_{transform}} \quad (7)$$

where S_{time} refers to the number of time sequences qualified in time domain while $S_{transform}$ is the number of time sequences qualified in the transformed domain. As we can observe, K-L transform gives the best precision at each dimension. On the other hand, the precision attained by Haar transform is close to the best and it outperforms DFT significantly at all except the first dimension. The enhancement in precision of Haar transform over DFT increases with the number of dimensions.

In Figure 4, the precision of Haar and Haar(V-shift) is shown. The precision of the non-v-shift model outperforms the v-shift model by 20% at most. The large difference can be attributed to the removal of the first Haar coefficient to achieve v-shift similarity. As the time series of financial data consist of a sequence of time values fluctuating around a relative constant level, which is the average value of that time sequence. This average value is very effective in discriminating time series in the sense that every sequence distributes further away in the x-axis. Hence, its removal will cause a sudden drop in precision. From another point of view, precision is traded for a better similarity model.

As most of the page accesses [6] of a query are devoted to removing false alarm, the precision is crucial to the overall performance of query evaluation. This agrees with the result depicted in Figure 5, where the page accesses of the best dimensions of DFT (dim. 5), Haar (dim. 7), and Haar(V-shift) (dim. 10) are shown. Page accesses increase linearly with ϵ. Haar has the minimum page accesses while DFT performs the worst. Page accesses of Haar(V-shift) model have been traded for better similarity model. Even so, it outperforms DFT. The best dimension of DFT is smaller than Haar and Haar(V-shift) as there is no significant gain in precision with additional dimensions. Haar(V-shift) needs more dimensions to attain sufficient precision in building the R-Tree.

Result of nearest neighbor query is shown in Figure 6. The trends for page accesses [7] are consistent with range query in Figure 5, Haar and Haar(V-shift) still outperform DFT.

Since many real data like stock movements and exchange rates can be modeled successfully by random walks [10], we also study the performance of our proposed technique for random walk data. Synthetic random walk data consisting of 30k time sequences are generated. As we want to show the effectiveness of our approach for different sequence lengths, we set ω = 1024. The same set of experiments as for the real data are performed and the results are found to be similar to that for the real data. The gain in performance by Haar is larger as longer sequences are used. For lack of space the details are not shown here.

[6] Performance is measured in terms of page access due to I/O time domination over computation time in database applications. Page accesses = non-leaf node accesses + leaf node accesses + post processing page accesses

[7] Page accesses = non-leaf node accesses + leave node accesses

5.1. Scalability Test

We study the scalability of our method by varying the size or length of synthetic time series database. Different sizes of databases (5k to 30k) and different lengths of sequences (256 to 2048) are generated as described in the previous section separately.

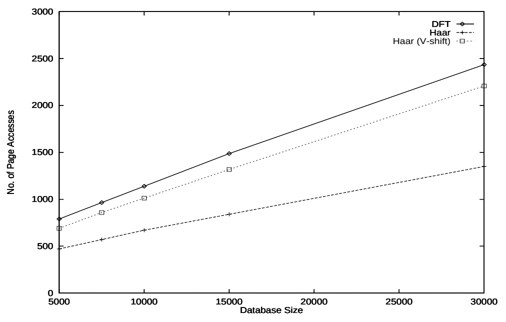


Figure 7. Scalability in database size of NN query

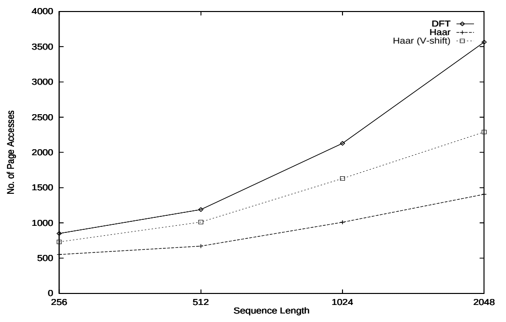


Figure 8. Scalability in sequence length of NN query

Figure 7 and Figure 8 show the scalability of nearest neighbor queries. Haar and Haar(V-shift) have a better scaling with database size and sequence length increase than DFT. Similar results have also been recorded for range queries. As revealed from the above experiments, a considerable portion of page accesses is devoted to the post processing step. The poorer precision of DFT creates more work in the post-processing step and this affects the overall performance, especially in terms of the amount of disk accesses for large databases with long sequences.

5.2. Other Wavelets

There are many kinds of known wavelets, we have tried some other wavelets in our experiments. We observe that Haar wavelets outperforms Daubechies and Coiflet wavelets in precision. Moreover, it is computationally less expensive. We discover that not all the wavelets are suitable for dimension reduction for stock data. From our experiments, not all the wavelets are able to concentrate energy at the first few coefficients. Haar, Daub4, and Coif6 are the best wavelets we have found in their families. From experiments, we find that the other wavelets seem to also preserve Euclidean distances, however, so far we have a proof of this property only for the Haar wavelets. It is interesting to see if we can apply different kinds of wavelets to different kinds of data series.

6. Conclusion

In this paper, an efficient time series matching technique through dimension reduction by Haar Wavelet Transform is proposed. The first few coefficients of the transformed sequences are indexed in an R-Tree for similarity search. Experiments show that our method outperforms the F-index (Discrete Fourier Transform) method in terms of pruning power, number of page accesses, scalability, and complexity. A new similarity model is introduced to deal with vertical shifts of sequences. Furthermore, an efficient two-phase nearest neighbor query is proposed and its effectiveness is demonstrated by experiments.

We have some suggestions for future work. We can study the possibility of using other wavelets like Symmlet [18] to boost up the performance further. We can also try to apply wavelets that did not work well with stock data in other signals, e.g. sinusoidal signals, electrocardiographs (ECGs).

References

[1] Johnson Ihyeh Agbinya. Discrete wavelet transform techniques in speech processing. *IEEE TENCON - Digital Signal Processing Applications*, pages 514–519, 1996.

[2] Rakesh Agrawal, Christos Faloutsos, and Arun swami. Efficient similarity search in sequence databases. In *Procs. of the Fourth International Conference on Foundations of Data Organization and Algorithms*, 1993.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. *IEEE*, pages 3–14, 1995.

[4] Ali N. Akansu and Richard A. Haddad. *Multiresolution Signal Decomposition*. Academic Press, 1992.

[5] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Procs. of ACM SIGMOD Conference on Management of Data*, pages 322–330, 1990.

[6] John J. Benedetto and Michael W. Frazier. *Wavelets – Mathematics and Applications*. CRC, 1994.

[7] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. An index structure for high-dimensional data. In *Procs. of the 22nd VLDB Conference*, 1996.

[8] Donald J. Berndt and James Clifford. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1995.

[9] C. Sidney Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms, A Primer*. Prentice Hall, 1997.

[10] C. Chatfield. *The Analysis of Time Series: an Introduction*. Chapman and Hall, 1984.

[11] King Lum Cheung and A. Fu. Enhanced nearest neighbor search on the R*-tree. *ACM SIGMOD Record, to appear*, Sept, 1998.

[12] Kam Wing Chu, S. K. Lam, and M. H. Wong. An efficient hash-based algorithm for sequence data searching. *The Computer Journal, to appear*, 1999.

[13] Tzi cker Chiuen. Content-based image indexing. In *Procs. of the 20th VLDB Conference*, pages 582–593, 1994.

[14] Tim Edwards. Discrete wavelet transforms: Theory and implementation. Technical report, Stanford University, 1991.

[15] Christos Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Procs. of the ACM SIGMOD Conference on Management of Data*, pages 419–429, 1994.

[16] Dennis Gabor. Theory of communication. *Journal of the Institute of Electrical Engineers*, 93(22):429–257, 1946.

[17] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison Wesley, 1992.

[18] Amara Graps. An introduction to wavelets. IEEE, 1995.

[19] K. Grochenig and W. R. Madych. Multiresolution analysis, haar bases, and self-similar tilings of r^n. *IEEE Trans. of Information Theory*, 38(2):556–568, 1992.

[20] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Procs. of the ACM SIGMOD Conference on Management of Data*, pages 47–57, 1984.

[21] Alfred Haar. Theorie der orthogonalen funktionen-systeme. *Mathematische Annalen*, 69:331–371, 1910.

[22] Flip Korn, H. V. Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Procs. of the ACM SIGMOD Conference on Management of Data*, 1997.

[23] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice Hall, 1975.

[24] Davood Rafiei and Alberto Mendelzon. Similarity-based queries for time series data. In *Procs. of the ACM SIGMOD Conference on Management of Data*, pages 13–25, 1997.

[25] Nick Roussopoulos, Stephen Kelley, and Frederic Vincent. Nearest neighbor queries. In *Procs. of ACM SIGMOD Conference on Management of Data*, pages 71–79, 1995.

[26] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann, 1996.

[27] Karl Weierstrass. *Mathematische Werke, Volume II*. Mayer & Muller, Berlin, 1895.

[28] Daniel Wu, D. Agrawal, A. E. Abbadi, A. Singh, and T. R. Smith. Efficient retrieval for browsing large image databases. In *Procs. Conf. on Information and Knowledge Management*, 1996.

Appendix A

The *Haar wavelets* are defined as

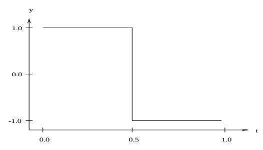
$$\psi_i^j(x) = \psi(2^j x - i) \qquad i = 0, \ldots, 2^j - 1 \tag{8}$$

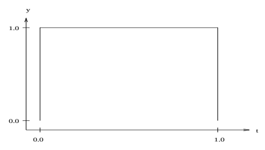
$$\text{where} \quad \psi(t) = \begin{cases} 1 & 0 < t < 0.5 \\ -1 & 0.5 < t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

together with a *scaling* function

$$\varphi(t) = \begin{cases} 1 & 0 < t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Haar wavelet for $\psi_0^0(t)$ and the scaling function are shown below.


y
1.0
0.0
-1.0
0.0
0.5
1.0
t


y
1.0
0.0
0.0
1.0
t