Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases

Eamonn Keogh[1], Kaushik Chakrabarti[2], Michael Pazzani[1] and Sharad Mehrotra[1]

[1]Department of Information and Computer Science
University of California, Irvine
California 92697, USA
[2]Department of Computer Science
University of Illinois at Urbana Champaign

Abstract. The problem of similarity search in large time series databases has attracted much attention recently. It is a non-trivial problem because of the inherent high dimensionality of the data. The most promising solutions involve first performing dimensionality reduction on the data, and then indexing the reduced data with a spatial access method. Three major dimensionality reduction techniques have been proposed, Singular Value Decomposition (SVD), the Discrete Fourier transform (DFT), and more recently the Discrete Wavelet Transform (DWT). In this work we introduce a new dimensionality reduction technique which we call Piecewise Aggregate Approximation (PAA). We theoretically and empirically compare it to the other techniques and demonstrate its superiority. In addition to being competitive with or faster than the other methods, our approach has numerous other advantages. It is simple to understand and to implement, it allows more flexible distance measures, including weighted Euclidean queries, and the index can be built in linear time.

Keywords: Time Series; Indexing and Retrieval; Dimensionality Reduction; Data Mining.

1. Introduction

Recently there has been much interest in the problem of similarity search in time series databases. This is hardly surprising given that time series account for much of the data stored in business, medical, and scientific databases. Similarity search is useful in its own right as a tool for exploring time series databases, and it is also an important subroutine in many KDD applications such as clustering [9], classification [18, 21] and mining of association rules [8].

Time series databases are often extremely large. Consider the MACHCO project. This astronomical database contains a half terabyte of data and grows at the rate of several gigabytes a day [21, 32]. Given the magnitude of many time series databases, much research has been devoted to speeding up the search process [1, 2, 3, 6, 11, 14, 17, 18, 19, 22, 23, 24, 30, 35]. The most promising methods are techniques that first perform dimensionality reduction on the data, and then use spatial access methods to index the data in the transformed space. The technique was introduced in [1] and extended in [11, 23, 24, 35]. The original work by Agrawal et al. utilizes the Discrete Fourier Transform (DFT) to perform the dimensionality reduction, but other techniques have been suggested, including Singular Value Decomposition (SVD) [34] and the Discrete Wavelet Transform (DWT) [6].

In this paper we introduce a novel transform to achieve dimensionality reduction. The method is motivated by the simple observation that for most time series datasets we can approximate the data by segmenting the sequences into equi-length sections and recording the mean value of these sections. These mean values can then be indexed efficiently in a lower dimensionality space. We theoretically and empirically compare our approach to all the other techniques, and demonstrate its superiority.

In addition to being competitive with or faster than the other transforms, we demonstrate that our approach has numerous other advantages. It is simple to understand and to implement, it allows for more flexible distance measures including the weighted Euclidean distance measure, and the index can be built in linear time. In addition our method also allows queries which are shorter than the query length for which the index was built. This very desirable feature is impossible in DFT, SVD and DWT due to translation invariance [29, 31].

The rest of the paper is organized as follows. In Section 2. we define the similarity search problem more formally and review GEMINI, a generic framework that utilizes any dimensionality reduction technique to index any kind of data [1, 8]. Establishing this generic framework allows us to compare the four dimensionality reduction techniques independent of any implementation decisions of the original authors. In Section 3. we introduce our method. In Section 4 we review the three other techniques and consider related work. Section 5 contains extensive empirical evaluation of all four methods. In Section 6. we demonstrate how our technique allows more flexible distance measures, including the weighted Euclidean distance. Section 7 offers concluding remarks and directions for future work.

2. Background

Given two sequences $X = x_1 \ldots x_n$ and $Y = y_1 \ldots y_m$ with $n = m$, their Euclidean distance is defined as:

$$D(X,Y) \equiv \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (1)$$

There are essentially two ways the data might be organized [8]:

- *Whole Matching.* Here it assumed that all sequences to be compared are the same length.
- *Subsequence Matching.* Here we have a query sequence *X*, and a longer sequence *Y*. The task is to find the subsequence in *Y*, beginning at Y_i, which best matches *X*, and to report its offset from Y_1.

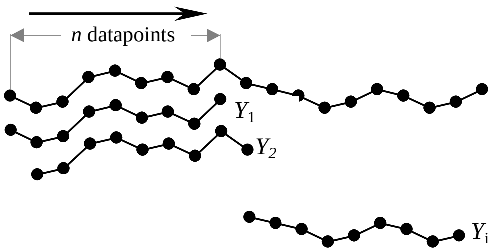


Fig 1. The subsequence matching problem can be converted into the whole matching problem by sliding a "window" of length *n* across the long sequence and making copies of the data falling within the windows

Whole matching requires comparing the query sequence to each candidate sequence by evaluating the distance function and keeping track of the sequence with the lowest distance. Subsequence matching requires that the query X be compared to every possible offset within the longer sequence Y. Note that it is possible to convert subsequence matching to whole matching by sliding a "window" of length n across Y, and making copies of the m-n windows. Figure 1 illustrates the idea. Although this causes storage redundancy, it simplifies the notation and algorithms, so we will adopt this policy for the rest of this paper.

There are two important kinds of queries that we would like to support in time series databases: *range queries* (e.g., return all sequences within an epsilon of the query sequence) and *nearest neighbor* (e.g., return the k closest sequences to the query sequence). The brute force approach to answering these queries, sequential scanning, requires comparing every time series X to Y_i. Clearly, this approach is unrealistic for large datasets.

Any indexing scheme that does not examine the entire dataset could potentially suffer from two problems, false alarms and false dismissals. False alarms occur when objects that appear to be close in the index are actually distant. Because false alarms can be removed in a post-processing stage (by confirming distance estimates on the original data), they can be tolerated so long as they are relatively infrequent. In contrast, false dismissals, when qualifying objects are missed because they appear distant in index space, are usually unacceptable. In this work, we will focus on admissible searching, indexing techniques that guarantee no false dismissals. Many inadmissible schemes have been proposed for similarity search in time series databases [2, 14, 19, 22, 30]. As they focus on speeding up search by sacrificing the guarantee of no false dismissals, we will not consider them further.

As noted by Faloutsos et al. [11], there are several highly desirable properties for any indexing scheme:

1) It should be much faster than sequential scanning.
2) The method should require little space overhead.
3) The method should be able to handle queries of various lengths.
4) The method should allow insertions and deletions without requiring the index to be rebuilt.
5) It should be correct, i.e. there should be no false dismissals.

We believe that there are two additional desirable properties:

6) It should be possible to build the index in "reasonable time".
7) The index should be able to handle different distance measures.

The sixth desirable property is introduced to exclude from consideration techniques such as Approximate Distance Maps [28]. This technique involves precomputing the distances between every pair of objects in the database to build a distance matrix, which becomes the index. The triangular inequality can then be used to prune most of the index during search. Modifications of the approach allow the larger values in the distance matrix to be discarded, so that by using a sparse matrix representation desirable property 2 is not violated. However, the approach has a time complexity quadratic in the number of items, which is simply intractable for even moderately large datasets.

The seventh requirement is motivated by the observation that for some applications the Euclidean distance measure can produce notions of similarity which are very unintuitive [18]. The "best" distance measure can depend on the dataset, task and user. It is even possible that the "best" distance measure to use could change as the user interactively explores a database [17]. Although the Euclidean distance is a good "gold standard" with which to compare different approaches, support for more flexible distance measure is desirable.

Sections 3 and 4 will evaluate the four dimensionality reduction techniques using these seven criteria.

2.1 Using dimensionality reduction for indexing

A time series *X* can be considered as a point in *n*-dimensional space. This immediately suggests that time series can be indexed by Spatial Access Methods (SAMs) such as the R-tree and its many variants [12]. However, most SAMs begin to degrade rapidly at dimensionalities greater than 8-12 [5], and realistic queries typically contain 20 to 1,000 datapoints. In order to utilize SAMs, it is necessary to first perform dimensionality reduction. In [11] the authors introduced the GEneric Multimedia INdexIng method (GEMINI) which can exploit any dimensionality reduction method to allow efficient indexing. The technique was originally introduced for time series, but has been successfully extended to many other types of data [20].

A crucial result in [11] is the authors' proof that in order to guarantee no false dismissals, the distance measure in the index space must satisfy the following condition:

$$D_{\text{index space}}(\mathrm{A,B}) \leq D_{\text{true}}(\mathrm{A,B}) \qquad (2)$$

This theorem is known as the lower bounding lemma or the contractive property. Given the lower bounding lemma, and the ready availability of off-the-shelf SAMs, GEMINI requires just the following three steps:

- Establish a distance metric from a domain expert (in this case Euclidean distance).
- Produce a dimensionality reduction technique that reduces the dimensionality of the data from *n* to *N*, where *N* can be efficiently handled by your favorite SAM.
- Produce a distance measure defined on the *N* dimensional representation of the data, and prove that it obeys $D_{\text{index space}}(\mathrm{A,B}) \leq D_{\text{true}}(\mathrm{A,B})$.

Table 1 contains an outline of the GEMINI indexing algorithm. All sequences in ***Y*** are transformed by some dimensionality reduction technique and are then indexed by the spatial access method of choice. The indexing tree represents the transformed sequences as points in *N*-dimensional space. Each point contains a pointer to the corresponding original sequence on disk.

```
Algorithm BuildIndex(Y,n);              // Y is the dataset,
                                        // n is the size of the window
for i = 1 to K                          // For each sequence to be indexed
    Yi ← Yi – Mean(Yi);                 // Optional: remove the mean of Yi
    Ȳi ← SomeTransformation(Yi);        // Any dimensionality reduction technique
    Insert Ȳi into the index with a pointer to Yi on disk;
end
```

Table 1: An outline of the GEMINI index building algorithm.

Note that each sequence has its mean subtracted before indexing. This has the effect of shifting the sequence in the y-axis such that its mean is zero, removing information about its offset. This step is included because for most applications the offset is irrelevant when computing similarity.

Table 2 below contains an outline of the GEMINI range query algorithm.

```
Algorithm RangeQuery(Q,ε)
  Project the query Q into the same feature space as the index.
  Find all candidate objects in the index within ε of the query.
  Retrieve from disk the actual sequences pointed to by candidates.
  Compute the actual distances, and discard false alarms.
```

Table 2: The GEMINI range query algorithm.

The range query algorithm is called as a subroutine in the K nearest neighbor algorithm outlined in Table 3.

```
Algorithm K_NearestNeighbor(Q,K)
  Project the query Q into the same feature space as the index.
  Find the K nearest candidate objects in the index.
  Retrieve from disk the actual sequences pointed to by candidates.
  Compute the actual distances and record the maximum, call it εmax.
  Issue the range query, RangeQuery(Q,εmax)
  Compute the actual distances, and choose the nearest K.
```

Table 3: The GEMINI nearest neighbor algorithm.

The efficiency of the GEMINI query algorithms depends only on the quality of the transformation used to build the index. The tighter the bound on $D_{\text{index space}}(A,B) \leq D_{\text{true}}(A,B)$ the better, ideally we would like $D_{\text{index space}}(A,B) = D_{\text{true}}(A,B)$. Time series are usually good candidates for dimensionality reduction because they tend to contain highly correlated features. In particular, datapoints tend to be correlated with their neighbors (a phenomenon known as autocorrelation). This observation motivated the dimensionality reduction technique introduced in this paper. Because datapoints are correlated with their neighbors, we can efficiently represent a "neighborhood" of datapoints with their mean value. In the next section we formally introduce and define this approach.

3. Piecewise Aggregate Approximation

3.1 Dimensionality reduction

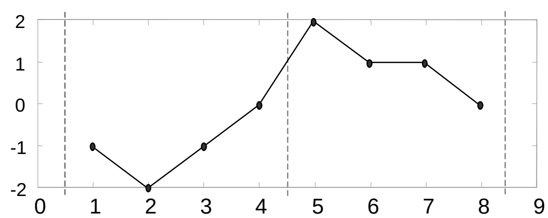
We denote a time series query as $X = x_1,\ldots,x_n$, and the set of time series which constitute the database as $\boldsymbol{Y} = \{Y_1,\ldots,Y_K\}$. Without loss of generality, we assume each sequence in $\boldsymbol{Y}$ is n units long. Let N be the dimensionality of the transformed space we wish to index

($1 \leq N \leq n$). For convenience, we assume that N is a factor of n. This is not a requirement of our approach, however it does simplify notation.

A time series X of length n is represented in N space by a vector $\overline{X} = \overline{x}_1, \ldots, \overline{x}_N$. The ith element of $\overline{X}$ is calculated by the following equation:

$$\overline{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j \tag{3}$$

Simply stated, to reduce the data from n dimensions to N dimensions, the data is divided into N equi-sized "frames". The mean value of the data falling within a frame is calculated and a vector of these values becomes the data reduced representation. Figure 2 illustrates this notation. The complicated subscripting in Eq. 3 just insures that the original sequence is divided into the correct number and size of frames.



X = (-1, -2, -1, 0, 2, 1, 1, 0) , $n = |X| = 8$

$\overline{X}$ = *(mean(-1,-2,-1,0), mean(2,1,1,0))*, N = | $\overline{X}$ | = *2*

$\overline{X}$ = (-1 , 1)

Fig 2. An illustration of the data reduction technique utilized in this paper. A time series consisting of eight (n) points is projected into two (N) dimensions. The time series is divided into two (N) frames and the mean of each frame is calculated. A vector of these means becomes the data reduced representation

Two special cases are worth noting. When $N = n$ the transformed representation is identical to the original representation. When $N = 1$ the transformed representation is simply the mean of the original sequence. More generally, the transformation produces a piecewise constant approximation of the original sequence. We therefore call our approach Piecewise Aggregate Approximation (PAA).

In order to facilitate comparison of PAA with the other dimensionality reduction techniques, it is useful to visualize it as approximating a sequence with a linear combination of box functions. Figure 3 illustrates this idea.

The time complexity for building the index for the subsequence matching problem appears to be O(nm), because for approximately m "windows" we must calculate Eq. 3 N times, and Eq. 3 has a summation of length n/N. However we can reduce the complexity to O(Nm) based on the following observation. Suppose we use Eq. 3 to calculate the features $\overline{x}_1, \ldots, \overline{x}_i, \ldots, \overline{x}_N$ for the first sliding window. Rather than use Eq. 3 again for the subsequent windows, we can simply adjust the features by subtracting out the influence of the datapoint pushed out the left of the frame, and factoring in the influence of the new datapoint arriving at the right of the frame. More concretely the i^{th} feature in j^{th} window can be updated as follows:

$$\overline{x}_{ji} = \overline{x}_{j-1i} - \frac{N}{n} x_{\frac{n}{N}(i-1)+1} + \frac{N}{n} x_{\left(\frac{n}{N}i\right)+1} \tag{4}$$

Consider Figure 2 as an example. The first feature extracted had its value calculated by Eq. 3 as –1 (the mean of –1, -2, -1 and 0). When we slide the window one unit to the right the four values used to determine the features value are now -2, -1, 0 and 2. Rather than use Eq. 3 again we can compensate for the –1 value lost and the 2 value gained by using Eq. 4 to calculate the new feature as $-\frac{1}{4} = -1 - \frac{2}{8}(-1) + \frac{2}{8}(2)$.

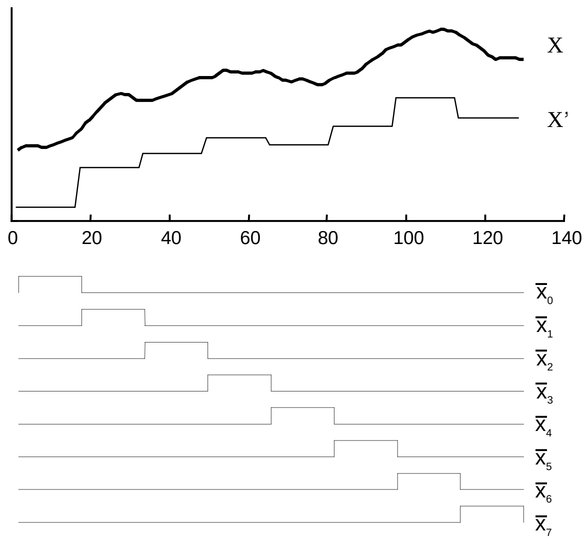


Fig 3. For comparison purposes it is convenient to regard the PAA transformation X', as approximating a sequence X with a linear combination of "box" basis functions

As mentioned in Section 2, in order to guarantee no false dismissals we must produce a distance measure *DR*, defined in index space, which has the following property: $DR(\overline{X},\overline{Y}) \le D(X,Y)$. The following distance measure has this property:

$$DR(\overline{X},\overline{Y}) \equiv \sqrt{\tfrac{n}{N}}\sqrt{\sum_{i=1}^{N}\left(\overline{x}_i - \overline{y}_i\right)^2} \tag{5}$$

The proof that $DR(\overline{X},\overline{Y}) \le D(X,Y)$ is straightforward but long. We relegate it to Appendix A to enhance the flow of the paper.

3.2 Handling queries of various lengths

In the previous section we showed how to handle queries of length *n*, the length for which the index structure was built. However, it is possible that a user might wish to query the index with a query that is longer or shorter than *n*. For example a user might normally be interested in monthly patterns in the stock market, but occasionally wish to search for weekly patterns. Naturally we wish to avoid building an index for every possible length of query. In this section we will demonstrate how we can execute queries of different lengths on a single fixed-length index. For convenience, we will denote queries longer than *n* as *XL* and queries shorter than *n* as *XS*, with $|XL| = n_{XL}$ and $|XS| = n_{XS}$.

3.2.1 Handling short queries

Queries shorter than *n* can be dealt with in two ways. If the SAM used supports dimension weighting (as does for example the hybrid tree [5]), one can simply weigh all the dimensions from $ceiling(\frac{N\, n_{XS}}{n})$ to *N* as zero. Alternatively, the distance calculation in Eq. 5 can have the upper bound of its summation modified so that:

$$Nshort = \left\lceil \frac{N n_{XS}}{n} \right\rceil, \qquad \sqrt{\tfrac{n}{N}}\sqrt{\sum_{i=1}^{Nshort}\left(\overline{x}_i - \overline{y}_i\right)^2} \tag{6}$$

The modification does not affect the admissibility of the no false dismissal condition in Eq.2. because the distance measure is the same as in Eq. 5 which we proved, except that here we are summing over an extra 0 to $\frac{n}{N}-1$ nonnegative terms on the larger side of the inequality. Apart from making either one of these changes, the search algorithms given in Tables 2 and 3 can be used unmodified.

3.2.2 Handling longer queries

Handling long queries is a little more difficult than the short query case. Our index only contains information about sequences of length n (projected into N dimensions) yet the query XL is of length n_{XL} with $n_{XL} > n$. However, we can regard the index as containing information about the prefixes of potential matches to the longer sequence. In particular we note that the distance in index space between the prefix of the query and the prefix of any potential match is always less than or equal to the true Euclidean distance between the query and the corresponding original sequence. Given this fact, the search algorithms given in Tables 2 and 3 can be used with two minor modifications.

1. In the first line of both algorithms, when the query is transformed into the representation used in the index, we need to replace X with $XL[1:n]$. The remainder of the sequence, $XL[n+1:n_{XL}]$, is ignored during this operation.
2. In the second line of both algorithms the original data sequence pointed most promising object in the index is retrieved. For long queries, the original data sequence retrieved and subsequently compared to XL must be of length n_{XL}, not n.

Although these minor modifications allow us to make long queries, we should expect the performance of the algorithm to degrade, because the tightness of the lower bound will decrease.

4. Alternative dimensionality reduction techniques and related work

There has been much work in dimensionality reduction for time series. This review is necessarily brief. We refer the interested reader to the relevant papers for a more detailed information.

4.1 Spectral decomposition (Discrete Fourier Transform)

The first technique suggested for dimensionality reduction of time series was the Discrete Fourier Transform (DFT) [1]. Since then there have been dozens of papers offering extensions and modifications [11, 23, 24, 35]. The basic idea of spectral decomposition is that any signal, no matter how complex, can be represented by the superposition of a finite number of sine (and/or cosine) waves, where each wave is represented by a single complex number known as a Fourier coefficient [29]. A time series represented in this way is said to be in the frequency domain. There are many advantages to representing a time series in the frequency domain; the most important of which for our purposes is data compression. A signal of length n can be decomposed into n sine/cosine waves that can be recombined into the original signal. However, many of the Fourier coefficients have very low amplitude and thus contribute little to reconstructed signal. These low amplitude coefficients can be discarded without much loss of information thereby producing compression.

The key observation is that the Euclidean distance between two signals in the time domain is preserved in the frequency domain. This result is an implication of a well-known result called Parseval's law [29]. Therefore if some coefficients are discarded then the estimate of the distance between two signals is guaranteed to be an underestimate

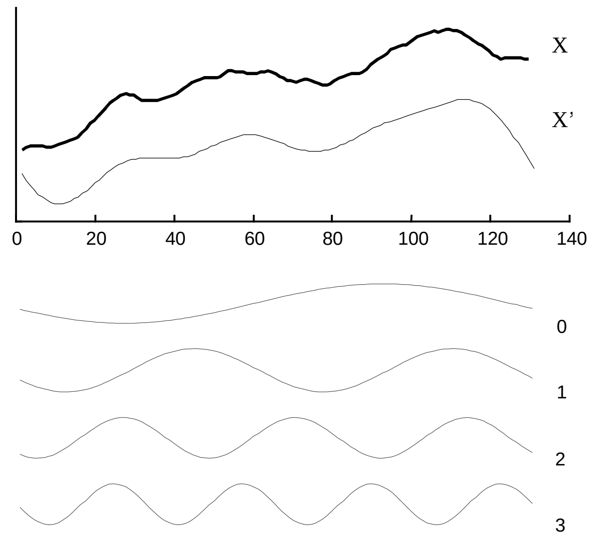


Fig 4. The first 4 Fourier bases can be combined in a linear combination to produce X', an approximation of the sequence X. Note that each basis wave requires two numbers to represent it (phase and magnitude)

(because some positive terms are ignored) thus obeying the lower bounding requirement in Eq. 2.

To perform the dimensionality reduction of a time series X of length n into a reduced feature space of dimensionality N, the Discrete Fourier Transform of X is calculated. The transformed vector of coefficients is truncated at $N/2$. The reason the truncation takes place at $N/2$ and not at N is that each coefficient is a complex number, and therefore we need one dimension each for the imaginary and real parts of the coefficients. Figure 4 illustrates this idea.

Given this technique to reduce the dimensionality of data from n to N, and the existence of the lower bounding distance measure, we can simply "slot" the DFT into the GEMINI framework.

The original work demonstrated a speedup of 3 to 100 over sequential scanning [1, 11]. However a very simple optimization first suggested in the context of images [34], and later independently rediscovered in the context of time series [23], allows even greater speedup. The optimization exploits a symmetric property of the DFT representation. In particular, the i^{th} +1 coefficient is always the complex conjugate of the n-i^{th} coefficient. So even though only N dimensions are retained in the index, an extra N "pseudo" dimensions can be used to estimate the distance between the query object and the candidates in the index, thus reducing the number of disk accesses. This optimization produces a further factor of two speedup [23]. We will denote this optimization as Symmetric DFT (SDFT) and utilize it for all experiments in this paper.

The time taken to build the entire index depends on the length of the sliding window. When the length is an integral power of two an efficient O($mn\log_2 n$) algorithm can be used. For other lengths a slower O(mn^2) algorithm must be employed.

There has been some work on more flexible distance measures in the frequency domain. Refiei and Mendelzon demonstrate a framework for handling several kinds of transformations such as moving average and time warping[1] [24]. However, because each coefficient represents a sine wave that is added along the entire length of the query it is not possible to use spectral methods for queries with "don't care" subsections [2] or the more general weighted Euclidean distance [17].

4.2 Wavelet decomposition

Wavelets are mathematical functions that represent data or other functions in terms of the sum and difference of a prototype function, called the analyzing or mother wavelet. In this sense they are similar to DFT. They differ in several important respects, however. One important difference is that wavelets are localized in time. This is to say that some of the wavelet coefficients represent small, local subsections of the data being studied. This is in contrast to Fourier coefficients that always represent global contributions to the data. This property is very useful for multiresolution analysis of the data. The first few coefficients contain an overall, coarse approximation of the data; addition coefficients can be imagined as "zooming-in" to areas of high detail. Figure 5 illustrates this. Recently there has been an explosion of interest in using wavelets for data compression, filtering, analysis, and other areas where Fourier methods have previously been used [4]. Not surprisingly, researchers have begun to advocate wavelets for indexing [20].

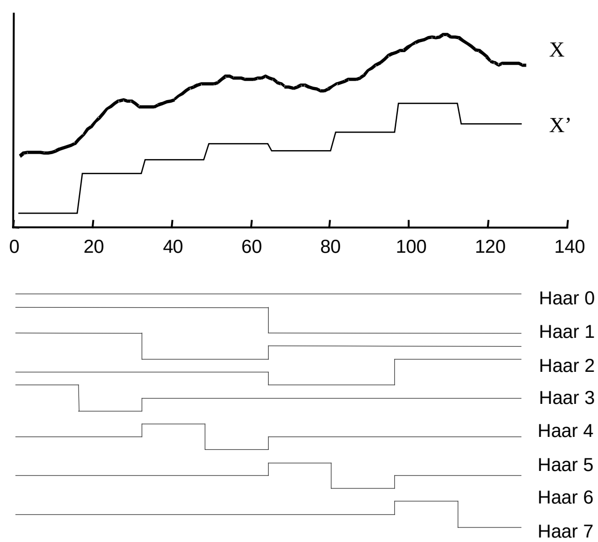


Fig 5. The first eight Haar wavelet bases can be combined in a linear combination to produce X', an approximation of the sequence X

Chan & Fu produced the breakthrough for wavelet indexing by producing a distance measure defined on wavelet coefficients which provably satisfies the lower bounding requirement in Eq. 2 [6]. The work is based on a simple, but powerful type of wavelet known as the Haar Wavelet. The Discrete Haar Wavelet Transform (DWT) can be calculated efficiently and an entire dataset can be indexed in O(*mn*).

[1] The transformation called "time warping" by Rafiei and Mendelzon is not the normal definition of the phrase. They consider only **global** scaling of the time axis.

DWT does have some drawbacks however. It is only defined for sequences whose length is an integral power of two. Although much work has been undertaken on more flexible distance measures using Haar [15, 31], none of these techniques are indexable.

The following result makes much of the discussion of the Haar wavelet irrelevant. The astute reader will have noticed that the PAA reconstruction of the sample time series shown in Figure 3 is identical to the Haar reconstruction of the same time series shown in Figure 5. As we will see, this is not a coincidence.

Theorem: If a set of time series is projected into a feature space with dimensionality that is a power of two, the representation of the Haar Transform and the representation of PAA are equivalent in the following ways:

1) The best possible reconstruction of the data using each approach is identical.
2) The estimated distances between any two objects in the feature space using either approach is identical.

The first item implies the second, because the tightest distance estimate between two objects should be the actual distance between the best reconstructions of the objects (by definition). The second point requires a very long but straightforward algebraic proof. For brevity, we will simply sketch an outline proof.

Consider the simple case where a time series of length n is represented by two PAA coefficients, P_0, P_1 or by two Haar coefficients, H_0, H_1. The two representations are related, $H_0 = (P_0 + P_1)/2$ and $H_1 = P_1 - P_0$. Using PAA, the best possible reconstruction of the first $n/2$ datapoints of the original time series is P_0 and the best possible reconstruction of the second $n/2$ datapoints is P_1. Using the Haar representation the best possible reconstruction of the two halves of the original time series are $H_0 - ½ H_1$ and $H_0 + ½ H_1$ respectively. We can substitute the identities above to get $(P_0 + P_1)/2 - ½ (P_1 - P_0)$ and $(P_0 + P_1)/2 + ½ (P_1 - P_0)$ which simplify to P_0 and P_1. Figure 6 illustrates this notation. A more general proof can be obtained by exploiting the recursive nature of both representations.

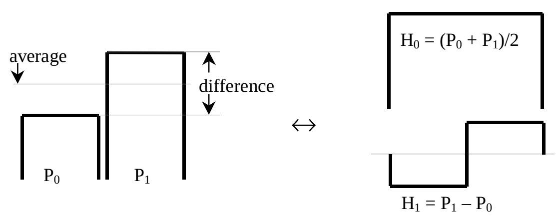


Fig 6. The PAA representation and the Haar representation are related. In particular, when the number of coefficients is a power of two it is always possible to convert from one representation to another. Here the two-coefficient case is illustrated

So PAA has all the pruning power of Haar, but is also able to handle arbitrary length queries, is much faster to compute, and can support more general distance measures. Note that we are not claiming that PAA is superior to Haar for all applications. But for the task of indexing PAA has all the advantages of Haar with none of the drawbacks. For completeness, we will still consider DWT in our experimental section.

4.3 Singular Value decomposition

Although Singular Value Decomposition (SVD) has been successfully used for indexing images and other multimedia objects [16, 34] and has been proposed for time series indexing [6], our paper contains the first actual implementation that the authors are aware of.

Singular Value Decomposition differs from the three other proposed techniques in an important respect. The other transforms are local; they examine one data object at a time and apply a transformation. These transformations are completely independent of the rest of the data. In contrast, SVD is a global transformation. The entire dataset is examined and is then rotated such that the first axis has the maximum possible variance, the second axes has the maximum possible variance orthogonal to the first, the third axis has the maximum possible variance orthogonal to the first two, etc. The global nature of the transformation is both a weakness and a strength from an indexing point of view.

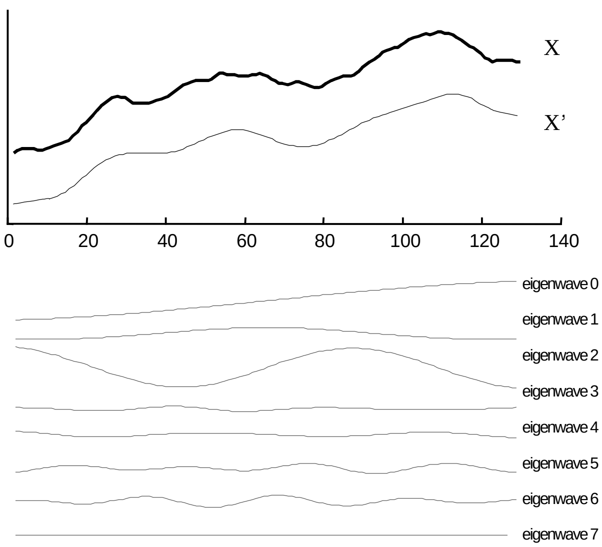


Fig 7. The first eight eigenwaves can be combined in a linear combination to produce X', an approximation of the sequence X

SVD is the optimal transform in several senses, including the following: If we take the SVD of some dataset, then attempt to reconstruct the data, SVD is the (linear) transform that minimizes reconstruction error [25]. Given this, we should expect SVD to perform very well for the indexing task.

SVD however, has several drawbacks as an indexing scheme, the most important of which relate to its complexity. The classic algorithms to compute SVD require $O(mn^2)$ time and $O(mn)$ space. Although linear in the number of data objects, the constants here are very high. Additionally, an insertion to the database requires recomputing the entire index. Fortunately, there has been much work recently on faster algorithms to compute SVD. Because of space limitations, we will not discuss the rival merits of the various approaches. We will simply use the EM approach suggested in [26]. This method has a sound theoretical basis and requires only $O(mnN)$ time and, more importantly for us, only $O(Nn + N^2)$ space. The problem of incremental updating has also received attention from many researchers. The fastest exact methods are still linear in m. Much faster approximate methods exist, but they introduce the possibility of false dismissals [7].

5. Experimental Results

5.1 Experimental methodology

We performed all tests over a range of dimensionalities and query lengths. We chose the range of 8 to 20 dimensions because that is the useful range for most spatial index structures [5, 13]. Because we wished to include the DWT in our experiments, we are limited to query lengths that are an integer power of two. We consider a length of 1024 to be the longest query likely to be encountered (by analogy, one might query a text database with a word, a phrase, or a complete sentence, but there would be little utility in a paragraph-length text query. A time series query of length 1024 corresponds approximately with sentence-length text query).

To perform realistic testing, we need queries that do not have exact matches in the database but have similar properties of shape, structure, spectral signature, variance, etc. To achieve this we used cross validation. We remove 10% of the dataset (a contiguous subsection), and build the index with the remaining 90%. The queries are then randomly taken from the withheld subsection. For each result reported on a particular dataset, for a particular dimensionality and query length, we averaged the results of 100 experiments.

Note that we only show results for nearest neighbor queries. However, in Table 2 we showed that the nearest neighbor algorithm invokes the range query algorithm as a subroutine, so we are also implicitly testing the range query algorithm.

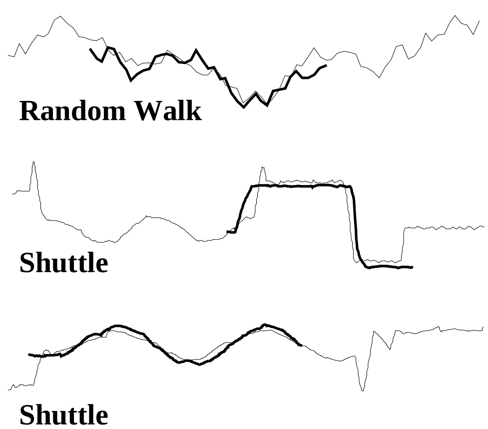


Fig 8. Sample queries (bold lines) and a section containing the corresponding best match for the two experimental datasets

We tested on two datasets with widely varying properties. Small, but typical subsections of each dataset are shown in Figure 8.

- **Random Walk:** The sequence is a random walk $x_t = x_{t-1} + z_t$ Where z_t (t = 1,2,…) are independent identically distributed (uniformly) random variables in the range (-500,500) [1].
- **Space Shuttle:** This dataset consists of 27 time series that describe the orientation of the Space Shuttle during the first eight hours of mission STS-57 [14, 15] (100,000 datapoints).

5.2 Experimental results: Building the index

We began our experiments by measuring the time taken to build an index for each of the suggested approaches. We did this for queries from length 32 to 1024 and database sizes of 40 to 640 kilobytes. The relatively small sizes of the databases were necessary to meaningfully include SVD in the experiments. Experimental runs that took more than 1,000 seconds were abandoned and are indicated by the black-topped histogram bars in Figure 9.We can see that SVD, using the classic algorithm, is simply intractable for even moderately sized databases. However the EM approach to SVD scales about as well as DFT or the DWT. The most obvious result from this experiment, however, is the celerity of PAA. For longer queries PAA, is many orders of magnitude faster than the other techniques.

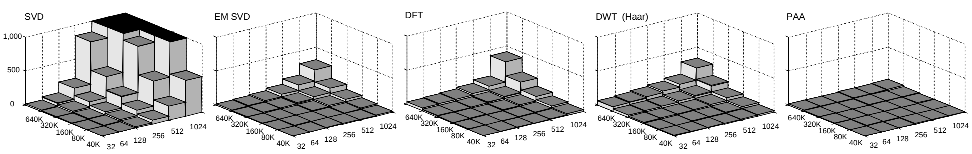


Fig 9: The time taken (in seconds) to build an index using various transformations over a range of query lengths and database sizes. The black topped histogram bars indicate that an experimental run was abandoned at 1,000 seconds

5.3 Experimental results: Pruning power

When comparing completing techniques there exists a danger of implementation bias. That is, consciously or unconsciously implementing the code such that some approach is favored. As example of the potential for implementation bias in this work consider the following. At query time the spectral approaches (DFT) must do a Fourier transform of the query. We could use the naïve algorithm which is $O(n^2)$ or the faster radix-2 algorithm (padding the query with zeros for $n \neq 2^{\text{integer}}$ [28]) which is $O(n\log n)$. If we implemented the simple algorithm it would make the other indexing methods perform better relative to DFT. While we do present detailed experimental evaluation of implemented systems in the next section, we also present experiments in this section which are free of the possibility of implementation basis. We achieve this by comparing the pruning power of the various approaches.

To compare the pruning power of the four techniques under consideration we measure *P*, the fraction of the database that must be examined before we can guarantee that we have found the nearest match to a nearest-neighbor query.

$$P = \frac{\text{Number of objects that must be examined}}{\text{Number of objects in database}} \tag{7}$$

To calculate *P*, we do the following. Random queries are generated (as described above). Objects in the database are examined in order of increasing (feature space) distance from the query until the distance in feature space of the next unexamined object is greater than the minimum actual distance of the best match so far. The number of objects examined at this point is the absolute minimum in order to guarantee no false dismissals.

Note that the value of *P* for any transformation depends only on the data and is completely independent of any implementation choices, including spatial access method, page size, computer language or hardware. A similar idea for evaluating indexing schemes appears in [13].

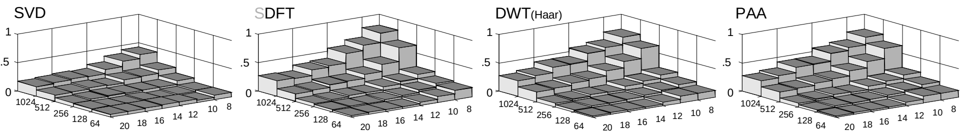


Fig 10. The fraction *P*, of a 64 megabyte Random-walk database that must be examined by the four dimensionality reduction techniques being compared, over a range of query lengths (2^6 to 2^{10}) and dimensionalities (8 to 20)

Figure 10 shows the value of *P* over a range of query lengths and dimensionalities. The experiment was conducted on 64 megabytes of the Random-Walk dataset. Note that the results for PAA and DWT are identical, as predicted in Section 4.2. DFT does not differ significantly from PAA for any combination of query length and dimensionality. For high dimensionalities/short queries, PAA is better than SVD. This might seem surprising given SVD's stated optimality; however it must be remembered that SVD's optimality only applies to the within sample data, and does not guarantee out of sample generalization optimality. However for more reasonable combinations of query lengths and dimensionalities, SVD is superior to PAA, but never by more than a factor of 2.3.

We repeated the experiment with databases of different sizes. As the size of the database gets larger, all techniques improve, that is to say, the value of *P* decreases. Figure 11 illustrates this. The amount of improvement may be difficult to discern from the graphs, however we note that all methods improve by the approximately the same amount, and thus all methods can be said to scale at the same rate.

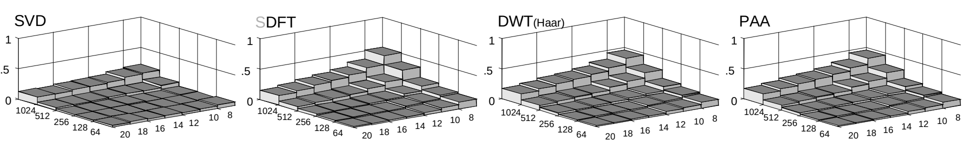


Fig 11. The fraction *P*, of a 256 megabyte Random-walk database that must be examined by the four dimensionality reduction techniques being compared, over a range of query lengths (2^6 to 2^{10}) and dimensionalities (8 to 20)

It is worthwhile to explain why the pruning power improves as the size of the database increases. Some researchers have observed this property and have claimed it as a property of their particular algorithm or representation. However, we believe the following explanation to be more credible.

The query algorithms terminate when the distance between the actual query and the best-match-so-far is less than the estimated distance between the query and the next closest match in the index. Therefore, the efficiency of the querying algorithms depends greatly on the quality of the eventual best match. A highly similar eventual best match allows the query algorithm to terminate earlier. A highly dissimilar eventual best match forces a larger subset of the database to be explored. In fact, the pathological case where the best match is an exact match to the query results in a constant time algorithm. Because we are experimenting with randomly generated data, the likelihood of a high quality match increases as a function of the database size, and thus we see this scaling effect.

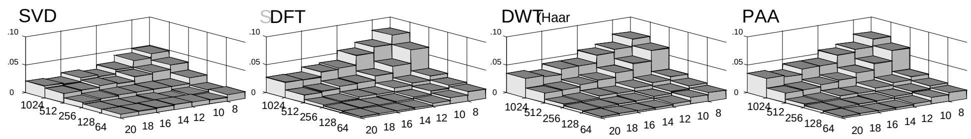


Fig 12. The fraction *P*, of Shuttle database that must be examined by the four dimensionality reduction techniques being compared, over a range of query lengths (2^6 to 2^{10}) and dimensionalities (8 to 20)

We also conducted similar experiments on the Shuttle dataset, the results are illustrated in Figure 12. This dataset has much more structure for the dimensionality reduction techniques to exploit, and all methods surpass their performance on the Random-Walk dataset (Note that the scale of the Z-axis in Figure 12 is different than in Figures 10 and 11). Overall, the relative performance of the different approaches is same as the previous experiments, except that the differences between the various techniques is even smaller.

5.4 Experimental results: Implemented system

Although the experiments in the previous section are powerful predictors of the (relative) performance of indexing systems using the various dimensionality reduction schemes, we also include a comparison of implemented systems for completeness.

All these experiments were conducted on a Sun Ultra Enterprise 3000 with 512 MB of physical memory and several GB of secondary storage. The spatial access method used was the Hybrid Tree [5].

To evaluate the performance of the four competing techniques we measured the normalized CPU cost.

Definition: *The Normalized CPU cost*: the ratio of average CPU time to execute a query using the index to the average CPU required to perform a linear (sequential) scan. The normalized cost of a linear scan is 1.0.

Using the normalized cost instead of direct costs allows us to compare each of the techniques against linear scan as the latter is widely recognized as a competitive search technique for high dimensional search spaces [5].

The experiments were conducted over a range of possible query-lengths, dimensionalities, and database sizes. For brevity we present just one typical result. Figure 13 shows the results of an experiment with a fixed query-length of 256 datapoints and a dimensionality of 16 with increasingly large datasets.

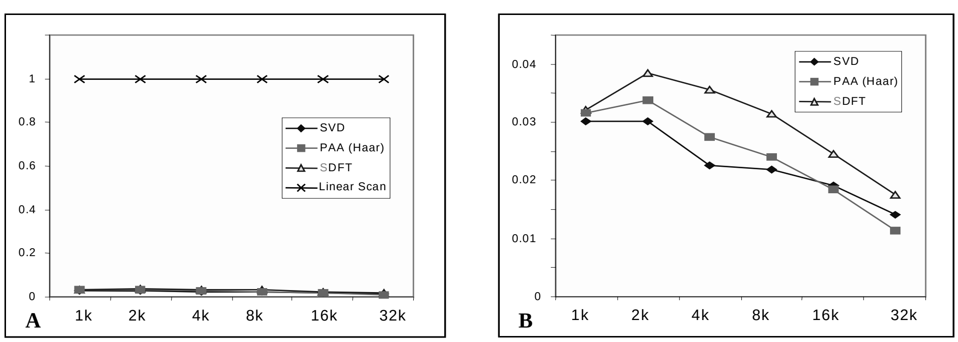


Fig 13. Scalability to database size. The X-axis is the number of objects in the database, the Y-axis is the normalized CPU cost. **A)** The normalized cost of the four dimensionality techniques compared to linear scan. **B)** A "zoom-in" showing the competing techniques in more detail

We can observe that all four techniques are orders of magnitude faster than sequential scanning, that there is relatively little difference between the competing techniques, and that difference decreases as the database grows larger. The overall conclusion of these experiments is that the choice of technique should not depend on speed-up, but on other factors such as the seven points enumerated in Section 2. As an example, in the next section we will show that PAA has an advantage over the other techniques in its ability to support more flexible distance measures.

6. Generalizing the Distance Measure

Although the Euclidean distance measure is optimal under several restrictive assumptions [1], there are many situations where a more flexible distance measure is desired [17]. The ability to use these different distance measures can be particularly useful for incorporating domain knowledge into the search process. One of the advantages of the indexing scheme proposed in this paper is that it can handle many different distance measures, with a variety of useful properties. In this section we will consider one very important example, weighted Euclidean distance. To the author's knowledge, this is the first time an indexing scheme for weighted Euclidean distance has been proposed.

6.1 Weighted Euclidean distance

It is well known in the Machine Learning community that weighting of features can greatly increase classification accuracy [32]. In [18] we demonstrated for the first time that weighting features in time series queries can increase accuracy in time series classification problems. In addition, in [17] we demonstrated that weighting features (together with a method for combining queries) allows relevance feedback in time series databases. Both [17, 18] illustrate the utility of weighted Euclidean metrics; however, no indexing scheme was suggested. We will now show that PAA can easily modified to support the weighted Euclidean distance.

In Section 3.2, we denoted a time series query as a vector $X = x_1,\ldots,x_n$. More generally we can denote a time series query as a tuple of equi-length vectors $\{X = x_1,\ldots,x_n$, $W = w_1,\ldots,w_n\}$ where X contains information about the shape of the query and W contains the relative importance of the different parts of the shape to the query. Using this definition the Euclidean distance metric in Eq. 1 can be extended to the weighted Euclidean distance metric DW:

$$DW([X,W],Y) = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2} \qquad (8)$$

We can perform weighted Euclidean queries on our index by making two simple modifications to the algorithm outlined in Table 2. We replace the two distance measures D and DR with DW and DRW respectively. DW is defined in Eq. 7 and DRW is defined as:

$$\overline{w}_i = \min(w_{\frac{n}{N}(i-1)+1},\ldots,w_{\frac{n}{N}i}), \qquad DRW([\overline{X},\overline{W}],\overline{Y}) = \sqrt{\tfrac{n}{N}}\sqrt{\sum_{i=1}^{N} \overline{w}_i (\overline{x}_i - \overline{y}_i)^2} \qquad (9)$$

Note that it is not possible to modify DFT or SVD in a similar manner, because each coefficient represents a signal that is added along the entire length of the query.

6.2 Experimental validation of the weighted Euclidean distance measure

To evaluate the effectiveness of the weighted-Euclidean distance measure, we generated a synthetic database of two similar time series. The database consists of 5,000 Class A and 5,000 Class B time series, which are defined as follows:

- **Class A**: $tan(sin(x^3))$, plus Gaussian noise with $\sigma = .2$ $-2 \le x \le 2$, 128 datapoints
- **Class B**: $sin(x^3)$, plus Gaussian noise with $\sigma = .2$ $-2 \le x \le 2$, 128 datapoints

Figure 14 shows an example of each class. Note that they are superficially very similar, although once it is observed that Class A has sharper peaks and valleys than Class B, humans can comfortably distinguish between the classes with essentially perfect precision.

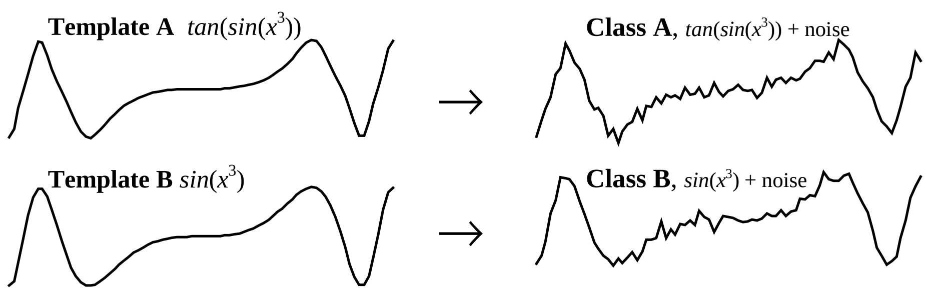


Fig 14. The synthetic data created for the for the Weighted Euclidean distance Vs Non-weighted Euclidean distance experiment. The two classes appear superficially similar, but are easily differentiated (by human visual inspection) after it is noted that Class A has "sharper" peaks and valleys than Class B.

The task for this experiment is as follows. Using the "template" (i.e. the shape before the noise is added) of a randomly chosen class as a query, retrieve the *K* most similar members of the same class from the database. We will compare two distance measures:

1) **WPAA:** The Weighted-Euclidean distance version of PAA defined in Equation 9 with a dimensionality of 16. We have the problem of how to set the weights. Given that we know the formula that produced the two classes and the fact that they are equiprobable, we could easily calculate the optimal weights. However to make the experiment more realistic, we set the weights by hand, simply assigning higher weights to the sections of the two classes that differ (i.e. the peaks and valleys) and lower values to the sections that are most similar (i.e. the broad center section). As we mentioned earlier, it would also be possible to learn the weights with relevance feedback in a way that is totally hidden from the user [17].
2) **EUCLID**: We want to have the best possible strawman to represent the non-weighted Euclidean distance metric. We considered using just the standard Euclidean metric defined in Equation 1. Using this metric we are ceding an advantage because this measure uses 8 times as much information as we are allowing WPAA (128 datapoints instead of 16). However, it might be argued that the inherent smoothing effect of dimensionality reduction using DFT, DWT, or SVD would help smooth out the noise and produce better results. Therefore, we ran all four approaches for every experimental run, and always chose the best one on that particular run. So we have **EUCLID** = *bestof*(Eq. 1, DFT, DWT, SVD).

We averaged the results of 100 experiments, building a new synthetic database for each run. We evaluated the effectiveness of each approach by measuring the average precision at the 1% to 10% percent recall points. Precision (P) is defined as the proportion of the returned sequences which are deemed relevant, and recall (R) is defined as the proportion of relevant items which are retrieved from the database. Figure 15 illustrates the results. We can see that the weighted Euclidean distance measure clearly outperforms the non-weighted Euclidean approach.

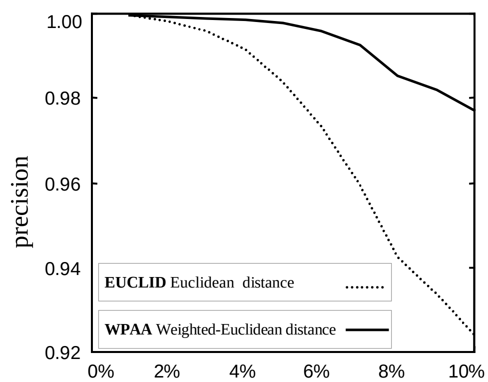


Fig 15. The mean precision of the two retrieval methods been compared, measured at recall points from 1% to 10%

7. Conclusions

The main contribution of this paper is to show that a surprisingly simple data transformation is competitive with more sophisticated transformations for the task of time series indexing. Additionally, we have demonstrated that the PAA transformation offers several advantages over competing schemes, including the following:

1) Being much faster to compute.
2) Being defined for arbitrary length queries (unlike DWT).
3) Allowing constant time insertions and deletions (unlike SVD).
4) Being able to handle queries shorter than that for which the index was built.
5) Being able to handle the weighted Euclidean distance measure.

It is the last item that we feel is the most useful feature of the representation, because it supports more sophisticated querying techniques like relevance feedback [17].

In future work we intend to further increase the speedup of our method by exploiting the similarity of adjacent sequences (in a similar spirit to the "trail indexing" technique introduced in [11]).

References

1. Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient similarity search in sequence databases. *Proc. of the 4th Conference on Foundations of Data Organization and Algorithms*.

2. Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in times-series databases. In *Proceedings of 21th International Conference on Very Large Data Bases*. Zurich. pp 490-50.

3. Bay, S. D. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

4. Bradley, C. Brislawn, and T. Hopper, "The FBI Wavelet/Scalar Quantization Standard for Gray-scale Fingerprint Image Compression," Tech. Report LA-UR-93-1659, Los Alamos Nat'l Lab, Los Alamos, N.M. 1993.

5. Chakrabarti, K & Mehrotra, S. (1999). The Hybrid Tree: An index structure for high dimensional feature spaces. *Proceedings of the 15th IEEE International Conference on Data Engineering*.

6. Chan, K. & Fu, W. (1999). Efficient time series matching by wavelets. *Proceedings of the 15th IEEE International Conference on Data Engineering*.

7. Chandrasekaran, S., Manjunath, B.S., Wang, Y. F. Winkeler, J. & Zhang. H. (1997) An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, Vol. 59, No. 5, pp. 321-332J.

8. Das, G., Lin, K. Mannila, H., Renganathan, G., & Smyth, P. (1998). Rule discovery from time series. In *Proceedings of the 3rd International Conference of Knowledge Discovery and Data Mining*. pp 16-22.

9. Debregeas, A. & Hebrail, G. (1998). Interactive interpretation of Kohonen maps applied to curves. *Proceedings of the 4th International Conference of Knowledge Discovery and Data Mining*. pp 179-183.

10. Faloutsos, C. & Lin, K. (1995). Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD Conf*., pp 163-174.

11. Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD Conf*., Minneapolis.

12. Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Conf*., pp 47-57.

13. Hellerstein, J. M., Papadimitriou, C. H., & Koutsoupias, E. (1997). Towards an analysis of indexing schemes. *Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*.

14. Huang, Y. W., Yu, P. (1999). Adaptive Query processing for time-series data. *Proceedings of the 5th International Conference of Knowledge Discovery and Data Mining*. pp 282-286.

15. Huhtala, Y., Kärkkäinen, J., & Toivonen. H. (1999) Mining for similarities in aligned time series using wavelets. *In Data Mining and Knowledge Discovery: Theory, Tools, and Technology*. SPIE Proceedings Series Vol. 3695., 150 - 160, Orlando, Florida.

16. Kanth, K.V., Agrawal, D., & Singh, A. (1998). Dimensionality reduction for similarity searching in dynamic databases. In *Proc. ACM SIGMOD Conf*., pp. 166-176.

17. Keogh, E. & Pazzani, M. (1999). Relevance Feedback Retrieval of Time Series Data. *Proc. of the 22th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.*

18. Keogh, E., & Pazzani, M. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. *Proceedings of the 4th International Conference of Knowledge Discovery and Data Mining*. pp 239-241, AAAI Press.

19. Keogh, E., & Smyth, P. (1997). A probabilistic approach to fast pattern matching in time series databases. *Proc. of the 3rd International Conference of Knowledge Discovery and Data Mining*. pp 24-20.

20. Korn, P., Sidiropoulos, N., Faloutsos, C., Siegel. E & and Protopapas. Z. (1996). Fast nearest-neighbor search in medical image databases. *Proceedings of 22th International Conference on Very Large Data Bases*, Bombay, India. pp 215-226.

21. Ng, M. K., Huang, Z., & Hegland, M. (1998). Data-mining massive time series astronomical data sets - a case study. *Proc. of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*. pp 401-402

22. Park, S., Lee, D., & Chu, W. (1999). Fast retrieval of similar subsequences in long sequence databases. *In 3rd IEEE Knowledge and Data Engineering Exchange Workshop*.

23. Rafiei, D. (1999) On similarity-based queries for time series data. *Proc of the 15th IEEE International Conference on Data Engineering*. Sydney, Australia.

24. Rafiei, D., & Mendelzon, A. (1997). Similarity-Based queries for time series data. In *Proc. ACM SIGMOD Conf*., pp. 13-25.

25. Ripley B.D. (1996) Pattern Recognition and Neural Networks. Cambridge University Press, 1996, ISBN 0-521-46086-7

26. Roweis, S,. (1998) EM Algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*. v.10, p.626, 1998.

27. Scargle, J. (1998). Studies in astronomical time series analysis: v. Bayesian blocks, a new method to analyze structure in photon counting data. Astrophysical Journal, Vol. 504.

28. Shasha, D., & Wang, T. L., (1990). New techniques for best-match retrieval. *ACM Transactions on Information Systems*, Vol. 8, No 2 April 1990, pp. 140-158.

29. Shatkay, H. (1995). The Fourier Transform - a Primer, Technical Report CS-95-37, Department of Computer Science, Brown University.

30. Shatkay, H., & Zdonik, S. (1996). Approximate queries and representations for large data sequences. *Proc. 12th IEEE International Conference on Data Engineering*. pp 546-553.

31. Struzik, Z. & Siebes, A. (1999). The Haar wavelet transform in the time series similarity paradigm. *In Proc 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*. pp 12-22.

32. Welch. D. & Quinn. P (1999)
http://www.macho.mcmaster.ca/Project/Overview/status.html

33. Wettschereck, D., Aha, D. & Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *AI Review*, Vol 11, Issues 1-5, pp. 273-314.

34. Wu, D., Agrawal, D., El Abbadi, A. Singh, A. & Smith, T. R. (1996). Efficient retrieval for browsing large image databases. *Proc of the 5th International Conference on Knowledge Information*. pp 11-18, Rockville, MD.

35. Yi, B,K., Jagadish, H., & Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. *IEEEE International Conference on Data Engineering*. pp 201-208.

Appendix

In the paper we claimed that the distance measure in the reduced space is always less than or equal to the Euclidean distance. We will now prove this. We present a proof for the case where there is a single frame. The more general proof for the *N* frame case can be obtained by applying the proof to each of the *N* frames.

As a preliminary we will define a vector of real numbers Δw. If w is a vector of real numbers and $\overline{w}$ is the arithmetic mean of w, then $\Delta w_i \equiv \overline{w} - w_i$. An important property of Δw is that it sums to zero.

Lemma: $\Sigma\Delta w_i = 0$. Both w and Δw are of length p. Let Σ denote the sum from 1 to p.

Assume	$\Sigma\Delta w_i = 0$
From the definition above	$\Sigma\, \overline{w} - w_i = 0$
Associative Law	$\Sigma\, \overline{w} - \Sigma\, w_i = 0$
$\overline{w}$ is a constant	$p\,\overline{w} - \Sigma\, w_i = 0$
Replace $\overline{w}$ with its definition	$p\,\frac{1}{p}\,\Sigma\, w_i - \Sigma\, w_i = 0$
Cancel p times its reciprocal	$\Sigma\, w_i - \Sigma\, w_i = 0$
Which completes the proof	$0 = 0$

The definition of Δw_i allows us to rewrite w_i as $\overline{w} - \Delta w_i$, a fact which we will utilize below.

Proof: Let *X* and *Y* be two time series, with $|X| = |Y| = n$. Let $\bar{x}$ and $\bar{y}$ be the corresponding transformed vectors defined in Eq. 3. We want to prove $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \geq \sqrt{\tfrac{n}{N}}\sqrt{\sum_{j=1}^{N}(\bar{x}_j - \bar{y}_j)^2}$

Because we are considering just the single frame case, we can remove summation over *N* frames and rewrite the inequality as:

Assume	$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \geq \sqrt{n}\sqrt{(\bar{x} - \bar{y})^2}$
Because $(AB)^x = A^xB^x$	$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \geq \sqrt{n(\bar{x} - \bar{y})^2}$
Because the terms under the radicals must be nonnegative, we can square both sides	$\sum_{i=1}^{n}(x_i - y_i)^2 \geq n(\bar{x} - \bar{y})^2$
Substitute rearrangement of definition above	$\sum_{i=1}^{n}((\bar{x} - \Delta x_i) - (\bar{y} - \Delta y_i))^2 \geq n(\bar{x} - \bar{y})^2$
Rearrange terms	$\sum_{i=1}^{n}((\bar{x} - \bar{y}) - (\Delta x_i - \Delta y_i))^2 \geq n(\bar{x} - \bar{y})^2$
Binomial theorem	$\sum_{i=1}^{n}(\bar{x} - \bar{y})^2 - 2(\bar{x} - \bar{y})(\Delta x_i - \Delta y_i) + (\Delta x_i - \Delta y_i)^2 \geq n(\bar{x} - \bar{y})^2$
Distributive law	$\sum_{i=1}^{n}(\bar{x} - \bar{y})^2 - \sum_{i=1}^{n}2(\bar{x} - \bar{y})(\Delta x_i - \Delta y_i) + \sum_{i=1}^{n}(\Delta x_i - \Delta y_i)^2 \geq n(\bar{x} - \bar{y})^2$
Summation properties	$n(\bar{x} - \bar{y})^2 - 2(\bar{x} - \bar{y})\sum_{i=1}^{n}(\Delta x_i - \Delta y_i) + \sum_{i=1}^{n}(\Delta x_i - \Delta y_i)^2 \geq n(\bar{x} - \bar{y})^2$
Associative law	$n(\bar{x} - \bar{y})^2 - 2(\bar{x} - \bar{y})\left(\sum_{i=1}^{n}\Delta x_i - \sum_{i=1}^{n}\Delta y_i\right) + \sum_{i=1}^{n}(\Delta x_i - \Delta y_i)^2 \geq n(\bar{x} - \bar{y})^2$
$\Sigma\Delta w_i = 0$, proved above	$n(\bar{x} - \bar{y})^2 - 2(\bar{x} - \bar{y})(0 - 0) + \sum_{i=1}^{n}(\Delta x_i - \Delta y_i)^2 \geq n(\bar{x} - \bar{y})^2$
Subtract like term from both sides	$n(\bar{x} - \bar{y})^2 + \sum_{i=1}^{n}(\Delta x_i - \Delta y_i)^2 \geq n(\bar{x} - \bar{y})^2$
The sum of squares must be nonnegative	$\sum_{i=1}^{n}(\Delta x_i - \Delta y_i)^2 \geq 0$
This completes the proof.	

Author Biographies

Eamonn Keogh received his B.S. from California State University, San Marcos, and his M.S from the Department of Information and Computer Science at the University of California, Irvine, where he is pursuing his Ph.D. His research interests include data mining, machine learning and information retrieval. He is a member of AAAI and ACM SIGKDD.

Kaushik Chakrabarti received his BTech degree in Computer Science and Engineering from the Indian Institute of Technology, Kharagpur in 1996 and M.S. degree in Computer Science from the University of Illinois at Urbana Champaign in 1999. He is currently pursuing his Ph.D. degree in Computer Science at the University of Illinois at Urbana Champaign. His research interests include multimedia databases, spatial and temporal databases, decision support systems, data mining, information retrieval in databases and web databases. He is a member of the Honor Society of Phi Kappa Phi and is a student member of the ACM, ACM SIGMOD and ACM SIGKDD.

Michael Pazzani is a Professor and former chair of the Information and Computer Science Department at the University of California, Irvine. He received his M.S. and B.S. in Computer Engineering from the University of Connecticut and his Ph.D. in Computer Science from UCLA. He is a member of AAAI and the Cognitive Science Society. His research interests include data mining and intelligent agents.

Sharad Mehrotra is an Associate Professor in the Information and Computer Science Department at the University of California, Irvine. He received his Ph.D. in Computer Science in 1993 and subsequently worked at MITL, Princeton as a scientist from 1993-1994. From 1994-1998 he worked as an assistant professor in the Computer Science department and at the Beckman Institute at the University of Illinois at Urbana-Champaign. He specializes in the areas of database management, distributed systems, and information retrieval. His current research is on multimedia databases, content-based retrieval, multidimensional indexing, uncertainty in databases, and spatio-temporal data management. He is an author of over 60 research publications in these areas. He is the recipient of the NSF Career Award and the Bill Gear Outstanding junior faculty research award at the University of Illinois at Urbana-Champaign in 1997.