Ensemble-Index: A New Approach to Indexing Large Databases

Eamonn Keogh
Dept of Info and Computer Science
University of California, Irvine
California 92697 USA
(949) 824-7210
eamonn@ics.uci.edu

Selina Chu
Dept of Info and Computer Science
University of California, Irvine
California 92697 USA
(949) 824-2968
selina@ics.uci.edu

Michael Pazzani
Dept of Info and Computer Science
University of California, Irvine
California 92697 USA
(949) 824-7405
pazzani@ics.uci.edu

ABSTRACT

The problem of similarity search (query-by-content) has attracted much research interest. It is a difficult problem because of the inherently high dimensionality of the data. The most promising solutions involve performing dimensionality reduction on the data, then indexing the reduced data with a multidimensional index structure. Many dimensionality reduction techniques have been proposed, including Singular Value Decomposition (SVD), the Discrete Fourier Transform (DFT), the Discrete Wavelet Transform (DWT) and Piecewise Polynomial Approximation. In this work, we introduce a novel framework for using ensembles of two or more representations for more efficient indexing. The basic idea is that instead of committing to a single representation for an entire dataset, different representations are chosen for indexing different parts of the database. The representations are chosen based upon a local view of the database. For example, sections of the data that can achieve a high fidelity representation with wavelets are indexed as wavelets, but highly spectral sections of the data are indexed using the Fourier transform. At query time, it is necessary to search several small heterogeneous indices, rather than one large homogeneous index. As we will theoretically and empirically demonstrate this results in much faster query response times.

Categories and Subject Descriptors

E.2 [**Data Storage Representations**]: H.3.1 Content Analysis and Indexing.

General Terms

Algorithms, Measurement, Design, Experimentation.

Keywords

Time series, indexing and retrieval, dimensionality reduction, similarity search, data mining.

1. INTRODUCTION

Recently there has been much interest in the problem of similarity search (query-by-content) in multimedia databases (i.e. time series, spatial data, images etc.). Similarity search is useful in its own right as a tool for exploratory data analysis, and it is also an important subroutine of many data mining applications such as clustering [8], classification [17] and mining of association rules [7]. The similarity between two objects can usually be calculated very efficiently, so the similarity search process is heavily I/O bound. The volume of data encountered exasperates the problem. Multi-gigabyte datasets are increasing prevalent. As typical example, consider the MACHCO project. This database contains more than a terabyte of data and is updated at the rate of several gigabytes a day [23].

The most promising similarity search methods are techniques that perform dimensionality reduction on the data, then use a multidimensional index structure to index the data in the transformed space. The technique was introduced in [1] and extended in [19, 6, 21, 10]. 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) [18, 15], the Discrete Wavelet Transform (DWT) [5, 24] and Piecewise Polynomial Approximations [15, 26].

In general, the efficiency of indexing depends only on the fidelity of the approximation in the reduced dimensionality space. Clearly no single dimensionality reduction technique can be optimal on all datasets, therefore, most researchers advocate empirical comparisons of several approaches before implementing a single fixed technique [24, 25, 15].

While this approach seems reasonable, it does risk the following problem. The chosen representation may tightly approximate most of the data, but provide a very poor approximation of a small section of the data. Any query for an item whose eventual best match is one of the poorly approximated items is likely to result in a long query time, because large amounts of the multidimensional index structure will have to be inspected. As an example, consider Figure 1. On a global level, this particular financial time series is much better approximated by the DFT transform than the DWT transform. However there are some small areas, for example beginning at about 850, where the DWT transform is superior. So, in general, we would be better off indexing this time series with DFT, but for queries that will have an eventual best match between 850 and 900, we would have been much better off had we built the index with the DWT instead.

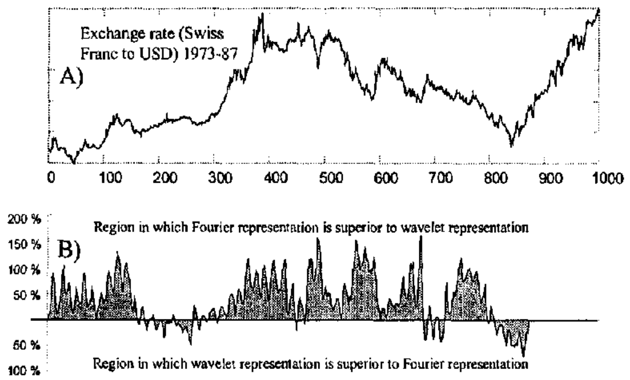


Figure 1. **A)** A financial time series. **B)** A graph showing the percentage by which the DFT is superior/inferior to DWT (calculated by {max(R_{DFT}, R_{DWT}) - min(R_{DFT}, R_{DWT})} / min(R_{DFT}, R_{DWT}) , where R is the reconstruction error for each approach). Note that while DFT is superior overall, there are small sections where DWT is up to 60% better.

More generally, it is possible that on a given dataset, two dimensionality reduction techniques have identical overall performance, yet on a local level, the two approaches have very different fidelity. As an example, consider Figure 2.

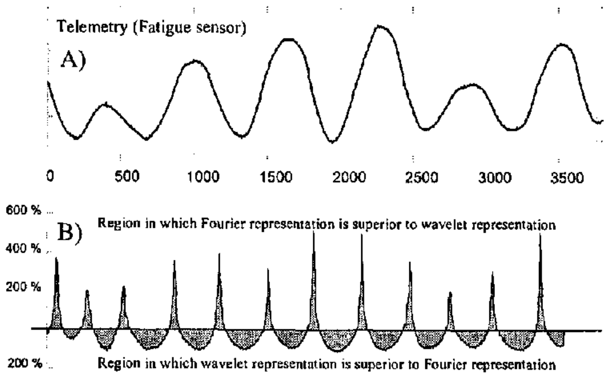


Figure 2. **A)** A scientific time series (Telemetry from a fatigue sensor). **B)** A graph showing the percentage by which the DFT is superior/inferior to DWT (calculated by {max(R_{DFT}, R_{DWT}) - min(R_{DFT}, R_{DWT})} / min(R_{DFT}, R_{DWT}), where R. is the reconstruction error for each approach). Note that while the average performance is almost identical, the two approaches have enormously differing abilities to represent the data on a local level.

These observations motivate the novel indexing approach introduced in this paper. We present a new framework called E-index (Ensemble-index). The basic idea to exploit the fact that different sections of the dataset favor different dimensionally reduction techniques. Rather than committing to a single dimensionally reduction technique, E-index creates several indices, one for each representation. When the data is indexed, E-index places each object in the index that can represent it with the greatest fidelity. At query time, the query is transformed into all the different representations in the ensemble, and the fidelity in each representation is measured. This information is then used to determine the order in which the indices are searched. E-index has a surprising and very unintuitive property. One might expect that its performance for any given query would be bound by the performance of the best possible single-representation index. In fact, as we will theoretically and empirically demonstrate, the performance of E-index can be better than the best possible single-representation index. In other words, an ensemble of K representations can outperform the best of those K representations used by itself, not only on average, but on *each* individual query.

Although the ideas presented in this paper apply to all types of multimedia data, for concreteness and brevity we will confine our discussion and experimental evaluation to time series data. Time series are an ubiquitous form of data, accounting for a large proportion of the data stored in financial, medical and scientific databases.

The rest of the paper is organized as follows. In Section 2 we provide background on indexing time series data for similarity search. In Section 3 we formally introduce E-index. Section 4 contains experimental evaluation. In Section 5 we consider some related work and finally in Section 6 we offer conclusions and directions for future work.

2. BACKGROUND

Given two time series Q = {q_1...q_n} and C = {c_1...c_n}, their Euclidean distance is defined as:

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2} \quad (1)$$

Figure 3 shows the intuition behind the Euclidean distance.

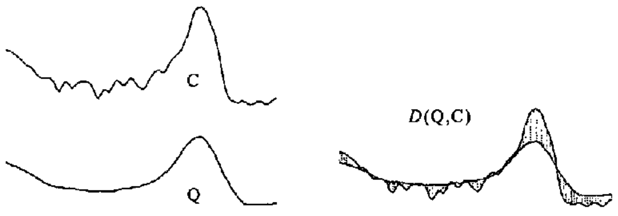


Figure 3. The intuition behind the Euclidean distance. The Euclidean distance can be visualized as the square root of the sum of the squared lengths of the gray lines.

There are essentially two ways the data might be organized [10]:

- *Whole Matching*. Here it assumed that all sequences to be compared are the same length.

- *Subsequence Matching*. Here we have a query sequence Q, and a longer sequence C of length *m*. The task is to find the subsequence in C, beginning at c_i, which best matches Q, and report its offset within C.

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 Q be placed at every possible offset within the longer sequence C. Note it is possible to convert subsequence matching to whole matching by sliding a "window" of length *n* across C, and making copies of the *m-n* windows. Figure 4 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.

The most important kind of query we would like to support are nearest neighbor queries (i.e. return the closest sequence to the query sequence). The brute force approach to answering these queries, sequential scanning, requires comparing every time series C_i to Q, is unrealistic for large datasets. Therefore a method for indexing the time series is desirable.

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. A false dismissal is when qualifying objects are missed because they appear distant in index space. In the next section we will discuss a widely used indexing technique that can utilize any dimensionality reduction technique and which guarantees no false dismissals.

2.1 The GEMINI Approach to Indexing

A time series $C = \{c_1 \ldots c_n\}$ with n datapoints can be considered as a point in n-dimensional space. This immediately suggests that time series could be indexed by multidimensional index structure such as the R-tree and its many variants [11]. Since realistic queries typically contain 20 to 1,000 datapoints (i.e. *n* varies from 20 to 1000) and most multidimensional index structures have poor performance at dimensionalities greater than 8-12 [12], we need to first perform dimensionality reduction in order to exploit multidimensional index structures to index time series data. In [10] the authors introduced GEneric Multimedia INdexIng method (GEMINI) which can exploit any dimensionality reduction method to allow indexing. The technique was originally introduced for time series, but has been successfully extended to other data types [18].

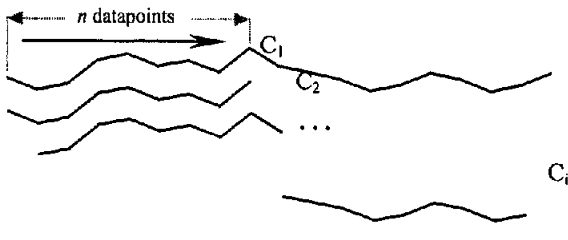


Figure 4: The subsequence matching problem can be converted into the whole matching problem by sliding a "window" of length *n* across the long sequence and making copies of the data falling within the windows.

An important result in [10] is that the authors proved that in order to guarantee no false dismissals, the distance measure in the index space must satisfy the following condition:

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

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 multidimensional index structures, 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 index structure.
- Produce a distance measure defined on the N dimensional representation of the data, and prove that it obeys

$$D_{\text{index space}}(A,B) \leq D_{\text{true}}(A,B).$$

Table 1 contains an outline of the GEMINI indexing algorithm. All sequences in the dataset C are transformed by some dimensionality reduction technique and 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.

Table 1. Outline of the GEMINI indexing building algorithm

```
Algorithm BuildIndex(C,n); // C is the dataset, n is the window size
for all objects in database
    $C_i \leftarrow C_i - \text{Mean}(C_i)$;          // Optional: remove the mean of $C_i$
    $\bar{C}_i \leftarrow \text{SomeTransformation}(C_i)$;
                          // $\bar{C}_i$ is any dimensionality reduced representation.
    Insert $\bar{C}_i$ into the SAM with pointer to $C_i$ on disk;
end;
```

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.

The nearest neighbor algorithm outlined in Table 2. For generality, instead of initializing `best-so-far` to infinity inside the algorithm, we can initialize it outside the algorithm, then pass it in. This generalization is just to make the outline of our algorithm simpler (cf. section 3).

Table 2. The GEMINI nearest neighbor algorithm

```
Algorithm NearestNeighbor(Q, best-so-far)
Let $\bar{Q}$ be the query Q, projected into the same feature space as the index.
while there are candidate objects closer to $\bar{Q}$ than best-so-far
  Find the candidate object $\bar{C}_i$ , nearest to $\bar{Q}$.
  Retrieve from disk the actual sequence pointed to by $\bar{C}_i$.
  Compute $D(Q,C_i)$, the distance between actual sequence and query.
  if $D(Q,C_i)$ < best-so-far
    best-so-far ← $D(Q,C_i)$
    index_of_NN ← i
  end;
end;
```

The efficiency of the GEMINI query algorithm depends only on the quality of the transformation used to build the index. The tighter the bound on $D_{index\ space}(A,B) \leq D_{true}(A,B)$ the better. Time series are usually good candidates for dimensionality reduction because they tend to contain highly correlated features.

This concludes the necessary background for this paper. For brevity, we will not describe the main dimensionality reduction techniques, SVD, DFT, DWT and PAA in detail. Instead we refer the interested reader to the relevant papers or to [15] which contains a survey of all the techniques.

3. INDEXING DATABASES WITH ENSEMBLES OF REPRESENTATIONS

In [24], and independently in [26], the authors presented a careful empirical comparison between DWT and DFT using the GEMINI framework and concluded that their average performance was nearly indistinguishable. Later, more general work, by the present authors confirmed this observation[1] [15]. However during a careful analysis of our results, we noted an interesting phenomenon. While it is true that on certain datasets DWT and DFT produce near identical indexing performance on *average*; On any individual query they can have very different performance. In other words, on any particular query, DWT can greatly outperform DFT or vice versa, it is only their mean behavior that is similar. For the rest of this work, we shall call the property of a set of different representations to have different indexing performance on *individual* queries VIP (Variability in Indexing Power).

This observation motivates our indexing algorithm. Assume that **R** is a set of K dimensionality reduction techniques proposed for indexing time series. Suppose we could produce a dimensionality reduction technique that performs as well as the best representation in **R** on *every* query. Then, averaged over many queries, the mean performance of our new approach must be at least as good as the average behavior of the best representation in **R**. More importantly, if there is significant VIP in the set of dimensionality reduction techniques in **R**, then the average performance of our new approach will be significantly better than the best technique in **R**.

Unfortunately, it is not possible to create a single dimensionality reduction technique that is as good as the best of a set of completing techniques on every query, however, we can do something almost equivalent. Instead of indexing the data with a single fixed representation, we can create an ensemble of indices. The intuition is that it is better to have many indices that specialize in representing certain kinds of data objects, than one general index that may represent some objects well but others poorly. When a query arrives, we convert it into each possible representation, and note the fidelity of each representation on that particular query. This information is then used to determine the order in which we search the indices.

It may not be obvious to the reader why this results in faster query response. After all, the total number of objects in the ensemble of indices that must be either examined or pruned, is the same as the number of objects that must be examined or pruned in the single representation approach. We will provide a worked example later to help develop the readers intuition. Until then, we offer this simple explanation. The number of objects that may be pruned depends upon the quality of the eventual best-match. So it is always to our advantage to find the best-match sooner rather than later. If the single representation approach poorly represents some objects, they may appear to be closer to the query than the eventual best-match, and thus we will be forced to retrieve them (i.e these items are false hits). In contrast, in the ensemble index, we do not have to deal with items in the index space that are poorly represented (they are in one of the other indices), so we have fewer false hits. This allows us to find a good match more quickly. Of course, we still have to deal with those objects we have avoided, but we can deal with them later, in a representation in which they are more tightly approximated. In addition, when we deal with them, we will already have found a good match, thus it is more likely that we will be able to prune them.

3.1 Algorithm descriptions

The algorithm used to build the E-index is described in Table 3. The inputs are **R**, the set of allowable representations (with $|\mathbf{R}| = K$) and C, the dataset containing *m* objects. The output is a set of K indices which taken together, index all *m* objects. Note the indices are not necessary all the same size (although we would prefer this). The distribution of sizes is determined by the data itself. Note we have no storage redundancy [12].

The algorithm used to search the E-index is described in Table 4. The input is the set of **K** indices and the query itself, and the output is the nearest neighbor match. The algorithm begins by transforming the query into each representation in the ensemble and recording its reconstruction error. This information is used to determine the order in which the indices are search. The indices are searched using the classic `NearestNeighbor` algorithm shown in Table 2, however there is one minor difference in how each search is initialized. The first index to be searched has its best-so-far argument initialized to infinity as usual. However, each subsequent invocation of `NearestNeighbor` inherits the best-so-far value from the previous invocation.

[1] More accurately, it was confirmed that DWT and DFT have similar performance on random walk data, and data that can be modeled by random walks (i.e stock market data). However, on many natural datasets, either of the techniques can dominate the other by orders of magnitude.

Table 3. The Ensemble-index building algorithm. The inputs are the dataset C, a list of K allowable representations R. The output is a set of K indices.

```
Algorithm BuildEnsembleIndex(C,R); // C = dataset
                                   // R = set of representations
for all objects in the database
  $C_i \leftarrow C_i - \text{Mean}(C_i)$;   // Optional: remove the mean of $C_i$
  for all possible representations $R_k$
    $\bar{C}_{ik} \leftarrow$ SomeTransformation($C_i$);
                      // $\bar{C}_{ik}$ is the object $C_i$ in the $K^{th}$
                      // dimensionality reduced representation.
    $RE_k \leftarrow D(\bar{C}_{ik}, C_i)$;     // Use Euclidean distance to
                                  // record the reconstruction error.
  end;
  index_to_use = $\text{argmin}RE_i \{RE_1,..,RE_K\}$
                                  // Record the best representation.
 Insert $\bar{C}_{index\_to\_use}$ into the $\text{index}_k$ with a pointer to $C_i$ on disk;
end;
```

Table 4. The E-index nearest neighbor algorithm

```
Algorithm Ensemble_NearestNeighbor(Q,R)
for all possible representations $R_k$
  $\bar{C}_{ik} \leftarrow$ SomeTransformation($C_i$);
                      // $\bar{C}_{ik}$ is the object $C_i$ in the $K^{th}$
                      // dimensionality reduced representation.
  $RE_k \leftarrow D(\bar{C}_{ik}$, Ci);     // Use Euclidean distance to
                              // record the reconstruction error.
end;
sort the indices in order of increasing reconstruction error.
  best-so-far = infinity;
for all indices                        // (In sorted order).
  best_match ← NearestNeighbor(Q, best-so-far)
                                               // ie. Table 2
  best-so-far ← D(Q, best_match)
                               //Use best-so-far from this search
end;                           // to seed the next search.
```

The speedup produced by the algorithm depends on the set of representations used and the data itself. Ideally we would prefer that the set of representations have high VIP with respect to the data. The set of representations available to the ensemble depends on the data type. For example, for indexing time series, the set of representations we could use include those listed in Table 5:

Table 5. Dimensionality reduction techniques that can be used with E-Index to index time series. Key: {} the introducing paper, [] extensions and follow up work.

- Discrete Fourier Transform {1}, [6, 10, 13, 18, 19, 21, 24].
- Discrete Wavelet Transform {5}, [13, 24].
- Piecewise Constant approximation {15}, [26].
- Piecewise Linear Approximation {17}, [22].
- Inner Product Approximation {9}.
- Adaptive Piecewise Constant Approximation {16}.

There are some notable omissions. We could include the Discrete Cosine Transform (DCT), however, because both DCT and DFT are spectral techniques we would expect very little VIP between them, and therefore there is very little point in including both. Singular Value Decomposition has been successfully implemented for time series [15] and other types of multimedia data [14] however it requires a global view of the data. It is possible that this list could be greatly expanded. The present authors have tested several novel representations in recent years. We were disappointed to find that while these representations work very well for certain types of data, they could not compete with existing approaches overall. This, of course, is exactly the property we desire for representations in the ensemble.

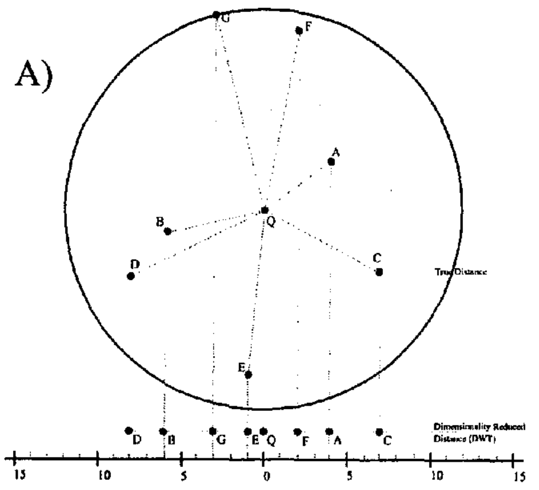
3.2 A Worked Example

The E-index has a very surprising and desirable property. Suppose we compare E-Index to standalone versions of indices using just one of the various representations in the ensemble. For example, we could compare E-Index with **R** = {DWT, DFT}, to both the classic DWT-Index and DFT-index. Given a query, E-Index can potentially return the best match faster than the best of the standalone approaches! Using the example above, it is possible that for a particular query, DWT-Index might require 100 disk accesses, DFT-Index might do better with only 90 disk accesses and E-Index might require only 50 disk accesses. Paradoxically, E-index can be better than the sum of its parts (more accurately, better than the best of it parts). This property is so unintuitive, we will provide a simple worked example to help the reader gain intuition for it.

Imagine we need to index a database with 7 objects {A, B, C, D, E, F, G}. We will trace the behavior of three approaches, DFT-Index, DWT-Index and E-Index with **R** = {DWT, DFT}, for a particular query Q. The estimated distances in the DFT space and DWT space, together with the true distances are shown in Table 6. Remember that the estimated distances can be obtained cheaply, by examining the index, but the true distances can only be obtained by expensive disk accesses, which we are trying to minimize.

Table 6. Column 1: The true distance between the query Q, and the seven candidate objects in the database. Columns 2 and 3: The estimated distances in the reduced dimensionality space for DFT and DWT. This information is graphically depicted in Figure 5.

	D_{true} (Q,O)	$D_{DFT}(\bar{Q},\bar{O})$	$D_{DWT}(\bar{Q},\bar{O})$
A	5	1	4
B	7	2	6
C	8	3	7
D	9	4	8
E	10	8	1
F	11	9	2
G	12	10	3



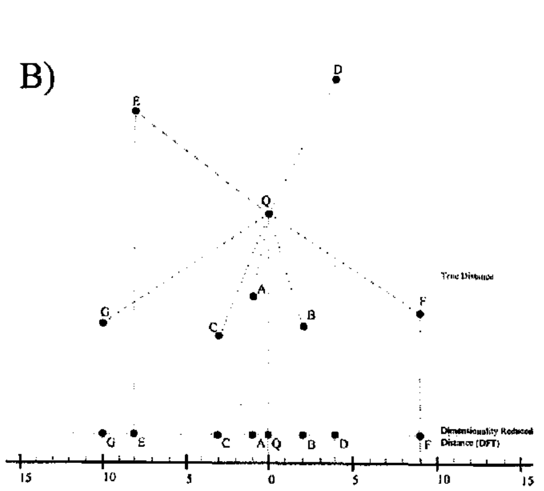


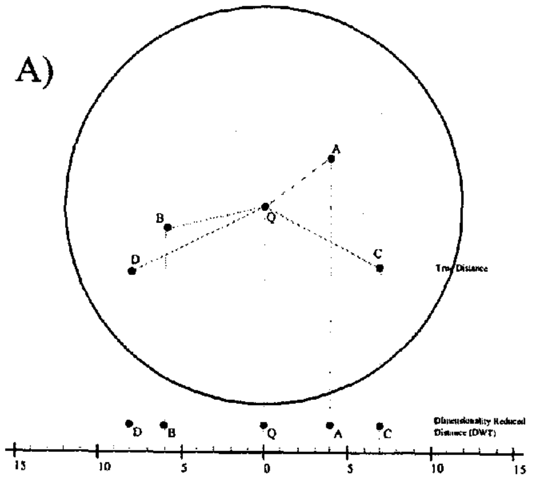
Figure 5. A graphical representation of the data in Table 6. The true distance between the query Q, and the seven candidate objects in the database is depicted in 2D space for both DWT (A) and DFT (B). Note that when the dimensionality is reduced from 2 dimensions to 1 dimension, the apparent distances in the reduced space are not accurate, but are lower bounding.

The DWT-Index is examined and the pointers to objects E, F, G, A, B, C, D are place in the priority queue, to be examined in that order. Object E is retrieved from disk, and found to have a true distance D_{true}(Q,E) of 10, so the best-so-far variable is initialized to 10. No objects can be pruned at this stage, so disk accesses are made to objects F, then G, then A. When A is retrieved, its true distance D_{true}(Q,A) is found to be 5, the best-so-far variable is therefore updated to 5. At that point, all the items left in the queue have lower bounds on their distance to the query which is greater than the best-so-far, so they can all be pruned. In summary, the DWT approach must make 4 disk accesses (to objects E, F, G and A).

The DFT-Index is examined and the pointers to objects A, B, C, D, E, F, G are place in the priority queue, to be examined in that order. Object A is retrieved from disk. It happens to be the best match, but it is not possible to tell that yet. The true distance D_{true}(Q,A) is 5, which means that objects E, F and G can all be pruned, because the lower bound on their distance to Q exceeds 5. One by one objects B, C and D are retrieved from disk and found to be no better than the current best-so-far. In summary, the DFT-Index must make 4 disk accesses (to objects A, B, C and D).

The E-Index builds two indices, a DWT-Index containing A, B, C, D and a DFT-Index containing E, F, G. The object A is retrieved from the disk, the true distance D_{true}(Q,A) is found to be 5. All the other items in the DWT-Index can be pruned because their distance to Q exceeds 5. The DFT-Index is visited next, all items in this index have a minimum distance that exceeds the best-so-far, so all objects can be pruned. In summary, the ensemble approach must make just 1 disk access (to object A).

This example is of course highly contrived, it is possible to produce an example in which E-Index does no better than its



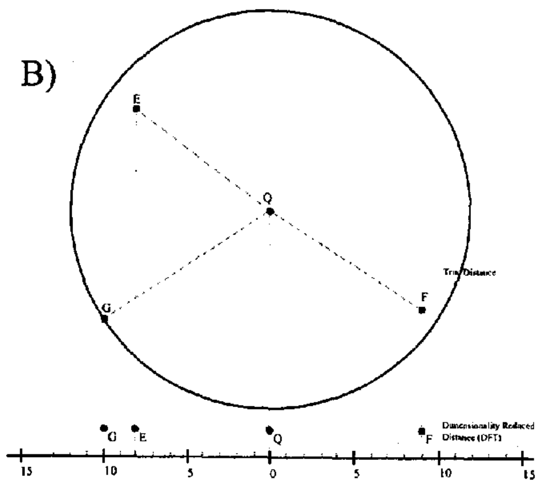


Figure 6. A graphical representation of the data in Table 6. Unlike Figure 5, only the objects that are tightly represented in wavelet space are placed in wavelet space (A), and only objects that are tightly represented in Fourier space are placed in Fourier space (B).

rivals. However it does illustrate the fact that an ensemble can be better than the best of it constituent parts. In the next section we will confirm this with an empirical study.

4. EXPERIMENTAL EVALUATION

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 removed 10% of the dataset (a contiguous subsection), and build the indexes 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 query length, we averaged the results of 1,000 experiments.

For simplicity and clarity we limit ensembles to size 2 (E-Index-2, **R** = {DWT, DFT}) and 3 (E-Index-3, **R** = {DWT, DFT, APCA}) and we compared them to standalone versions of DWT, DFT and APCA.

We tested on two datasets with widely varying properties. Many other researchers have used these datasets [1, 15, 17, 24, 26]. Small, typical subsections of each dataset are shown in Figure 7.

• **Random Walk**: The sequence is a random walk, xt = xt-1 + zt Where zt (t = 1,2,...) are independent identically distributed (uniformly) random variables in the range (-500,500).

• **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 (100,000 datapoints).

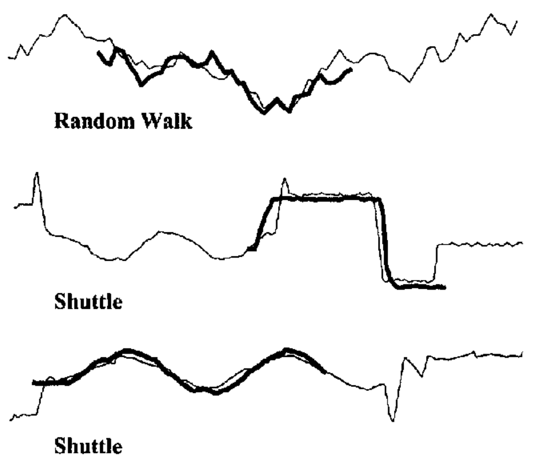


Figure 7. Sample queries (bold lines) and a section containing the best match for both the datasets used in the experimental study.

To compare the pruning power of the different 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 1-NN query.

$$P = \frac{\textit{Number of objects that must be examined}}{\textit{Number of objects in database}} \qquad (3)$$

Note the value of P for any indexing approach depends only on the efficiency of the indexing algorithm and the data itself. It 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 [18].

Table 7. The fraction *P*, of dataset that must be retrieved from disk using the five indexing approaches, for various dataset / query length combinations (averaged over 1,000 queries). The dimensionally of the index was 16 in every case.

Data/Query Length	DWT	DFT	APCA	E-Index-2	E-Index-3
Random Walk 512	0.31	0.24	0.32	0.19	0.19
Random Walk 1024	0.47	0.43	0.51	0.38	0.37
Space Shuttle 512	0.023	0.021	0.016	0.011	0.006
Space Shuttle 1024	0.041	0.039	0.022	0.027	0.010

The results for the random walk dataset are not spectacular, but the ensembles do outperform the three competing approaches. In fact, we anticipated this result because there is very little VIP in this domain between the 3 representations under consideration (See Appendix A). We hoped for better results with the Space Shuttle data because there is high VIP between the 3 representations for this data. Our optimism was justified, the Ensemble approach significantly outperforms the single representation approaches. We further note that the empirical results validate the prediction that the more representations added to ensemble the greater the increase in performance.

5. RELATED WORK

To the best of our knowledge, this paper is the first to suggest indexing data with ensembles of representations in order to improve the query response time. Others have suggested combining several representations in multimedia databases but their motivation was to provide more accurate matches in domains where the best match is more subjective [20]. The idea of using multiple representations to improve accuracy is also well understood in the text retrieval community [2]. Although they are using multiple representations to improve effectiveness and we are using multiple representations to improve efficiency, the underlying reason for the improvement in both cases is the same. Representations with globally similar performance may differ greatly on a local level. We exploit this fact by first querying the data in the representation to which it is best suited. Others exploit this fact by querying the data in *all* representations, and combining the results.

There has been much research in exact similarly search (i.e. search with the guarantee of no false dismissals) in time series [1, 5, 6, 9, 10, 15, 16, 19 21, 24, 26], and there is an even greater body of research in approximate search (i.e. search that may allow false dismissals) [22, 17]. For brevity we will not discuss this work here, instead we refer the reader to [15] or [16] which contains detailed discussions and extensive bibliographies.

6. CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

In this paper we introduce a new framework for indexing multimedia databases with ensembles of representations.

Directions for future work include:

- A more detailed theoretical analysis of our approach.
- A more extensive empirical study, including other types of multimedia data and using more representations in the ensemble.

We also believe that we may be able to further improve performance with a better implementation of the `Ensemble_NearestNeighbor` algorithm. We are currently relying on a heuristic to order the search of the representations in the ensemble. If possible, it would be better to search them in parallel, allowing all indices to share a single priority queue. There are, however, several difficult implementation issues that must be addressed before this is practical.

7. ACKNOWLEDGMENTS

The authors would like to thank Sharad Mehrotra and Michael Ortega-Binderberger for their helpful comments and suggestions.

8. REFERENCES

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

[2] Belkin, N., Cool, C., Croft, B & Callan, J. (1993). The effect of multiple query representations on information retrieval system performance. In Proceedings of the 16th ACM SIGIR Conference on Research and Development in Information Retrieval, pp 339--346.

[3] Chakrabarti, K & Mehrotra, S (2000). Local dimensionality reduction: A new approach to indexing high dimensional spaces. Proceedings of the 26th Conference on Very Large Databases.

[4] Chakrabarti, K., Ortega-Binderberger, M., Porkaew, K & Mehrotra, S. (2000) Similar shape retrieval in MARS. Proceeding of IEEE International Conference on Multimedia and Expo.

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

[6] Chu, K & Wong, M. (1999). Fast time-series searching with scaling and shifting. Proceedings of the 18th ACM Symposium on Principles of Database Systems, Philadelphia.

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

[8] 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.

[9] Egecioglu, O., & Ferhatosmanoglu, H. (2000). Dimensionality reduction and similarity distance computation by inner product approximations. Proceedings of the 9th ACM International Conference on Information and Knowledge Management (CIKM), pp. 219-226.

[10] Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In Proceedings of the SIGMOD.

[11] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. Proceedings ACM SIGMOD Conference. pp 47-57.

[12] Hellerstein, J. M., Papadimitriou, C. H., & Koutsoupias, E. (1997). Towards an analysis of indexing schemes. 16th ACM Symposium on Principles of Database Systems.

[13] Kahveci, T. & Singh, A (2001). Variable length queries for time series data. Proceedings 17th International Conference on Data Engineering. Heidelberg, Germany.

[14] Kanth, K.V., Agrawal, D., & Singh, A. (1998). Dimensionality reduction for similarity searching in dynamic databases. Proceedings ACM SIGMOD Conf., pp. 166-176.

[15] Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra (2001) Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems. Volume 3, Number 3, August.

[16] Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra (2001) Locally adaptive dimensionality reduction for similarity search in large time series databases. SIGMOD pp 151-162

[17] 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.

[18] Korn, F., Jagadish, H & Faloutsos. C. (1997). Efficiently supporting ad hoc queries in large datasets of time sequences. Proceedings of SIGMOD, Tucson, AZ, pp 289-300.

[19] Loh, W., Kim, S & Whang, K. (2000). Index interpolation: an approach to subsequence matching supporting normalization transform in time-series databases. Proceedings 9th International Conference on Information and Knowledge Management.

[20] Minka, T & Picard, R (1996) Interactive learning using a "society of models". In Proceedings IEEE Conference.on Computer Vision and Pattern. Recognition.

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

[22] Wang, C. & Wang, S. (2000). Supporting content-based searches on time Series via approximation. International Conference on Scientific and Statistical Database Management.

[23] Welch. D. & Quinn. P (1999). http://wwwmacho.mcmaster.ca/Project/Overview/status.html

[24] Wu, Y., Agrawal, D. & Abbadi, A.(2000). A Comparison of DFT and DWT based Similarity Search in Time-Series Databases. Proceedings of the 9th International Conference on Information and Knowledge Management.

[25] 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.

[26] Yi, B,K., & Faloutsos, C.(2000). Fast time sequence indexing for arbitrary Lp norms. Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt.

APPENDIX A

In this paper we informally defined VIP (Variability in Indexing Power) as the property of a set of different representations to have different performance on individual queries. Since the relative performance between two representations depends both on the indexed data and the query itself, we have postponed the challenge of formalizing a tighter definition or metric for future work. However we noted that we should expect VIP to be very highly correlated with the relative abilities of the different representations to compress the data. To test the validity of this assumption, we visualized the compressibility of two datasets in similar fashion to Figures 1 and 2.

The results, shown in Figure A1, seem to confirm the hypotheses. The three representations show very little local variability in their ability to compress the Random Walk dataset, and this result is echoed the relatively small speedup achieved by E-index. In contrast the three representations show great local variability in their ability to compress the Space Shuttle dataset, and in this dataset we saw the greatest speedup achieved by E-index. In future work we hope to exploit this relationship to formalize the notion of VIP.

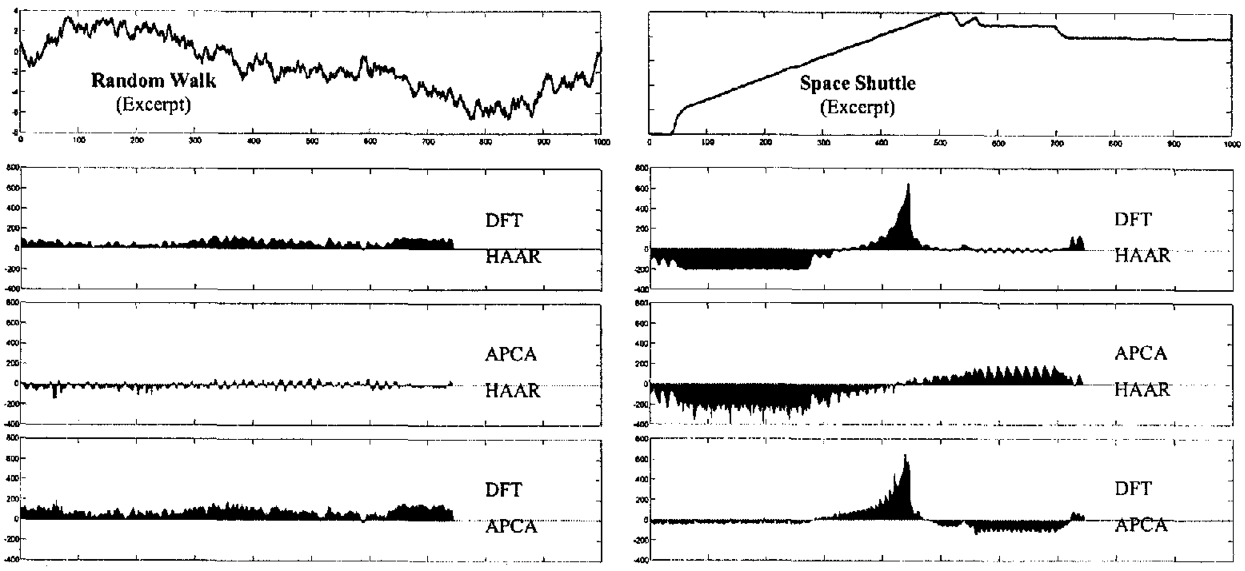


Figure A1. A series of graphs showing the percentage by which the various representations used in the experimental section are superior/inferior to each other (calculated by {max(RA, RB) - min(RA, RB)} / min(RA, RB) , where R is the reconstruction error for each approach). Note that there is little difference in the Random Walk domain, but large variance in the Space Shuttle domain.