



High dimensional nearest neighbor searching[☆]

Hakan Ferhatosmanoglu^{a,*}, Ertem Tuncel^b, Divyakant Agrawal^c,
Amr El Abbadi^c

^aComputer Science and Engineering, Ohio State University, 2015 Neil Ave, 395 Dreese Lab, Columbus, OH 43210, USA

^bElectrical Engineering, University of California, Riverside, USA

^cComputer Science, University of California, Santa Barbara, USA

Received 29 April 2004; received in revised form 11 September 2004; accepted 11 December 2004

Abstract

As databases increasingly integrate different types of information such as time-series, multimedia and scientific data, it becomes necessary to support efficient retrieval of multi-dimensional data. Both the dimensionality and the amount of data that needs to be processed are increasing rapidly. As a result of the scale and high dimensional nature, the traditional techniques have proven inadequate. In this paper, we propose search techniques that are effective especially for large high dimensional data sets. We first propose VA⁺-file technique which is based on scalar quantization of the data. VA⁺-file is especially useful for searching exact nearest neighbors (NN) in non-uniform high dimensional data sets. We then discuss how to improve the search and make it progressive by allowing some approximations in the query result. We develop a general framework for approximate NN queries, discuss various approaches for progressive processing of similarity queries, and develop a metric for evaluation of such techniques. Finally, a new technique based on clustering is proposed, which merges the benefits of various approaches for progressive similarity searching. Extensive experimental evaluation is performed on several real-life data sets. The evaluation establishes the superiority of the proposed techniques over the existing techniques for high dimensional similarity searching. The techniques proposed in this paper are effective for real-life data sets, which are typically non-uniform, and they are scalable with respect to both dimensionality and size of the data set.

© 2005 Elsevier B.V. All rights reserved.

Keywords: High dimensional data; Nearest neighbor queries; Indexing; Similarity search; Approximate and progressive search; Non-uniform data; Scalability; Performance

[☆]Recommended by N. Koudas, Area Editor.

*Corresponding author. Tel.: +1 614 2926377.

E-mail addresses: hakan@cse.ohio-state.edu (H. Ferhatosmanoglu), ertem@ee.ucr.edu (E. Tuncel), agrawal@cs.ucsb.edu (D. Agrawal), amr@cs.ucsb.edu (A. El Abbadi).

1. Introduction

As modern databases increasingly integrate various types of information, such as multimedia data, it becomes necessary to support efficient

retrieval in such systems. Example of such applications include multimedia information systems [1], CAD/CAM [2], geographical information systems (GIS) [3], time-series databases [4], medical imaging [5]. The data is usually represented by a feature vector which summarizes the original data with some number of dimensions. The similarity between two objects is defined with a distance function, e.g., Euclidean distance, between the corresponding feature vectors. A well-known type of query is the similarity query. For example, in image databases, the user may pose a query asking for the most similar images to a given image. Similarity query with multi-dimensional data is usually implemented by finding the k closest feature vectors to the feature vector of the query data, which is known as k -nearest neighbor (k -NN) query. A closely related query is the ε -range query where all feature vectors that are within ε neighborhood of the query point q are retrieved.

With the proliferation of multimedia, time-series, and scientific data, several applications need to be supported for indexing and retrieval of high dimensional data. In some applications, such as GIS, the feature vectors usually have small number of dimensions, typically 2 dimensions. Numerous index structures exist that facilitate search and retrieval of two or relatively low dimensional data [6–11]. The general approach for high dimensional indexing was to extend these spatial index structures and to propose new ones to deal with the high dimensional nature of information [12–14]. In fact, Berchtold et al. [15] and Weber et al. [16] have argued that using these multi-dimensional index structures for searching beyond a certain dimension becomes worse than a sequential scan. Weber et al. [16] have developed a quantitative analysis and performance study of similarity search techniques for high dimensional data sets. They formally show that for data sets with *uniform distribution*, the indexing techniques based on partitioning are outperformed on average by a simple sequential scan if the number of dimensions exceeds around 10. A new technique, called VA-file [16], has been shown to perform better than the current approaches, and to scale well with the dimensionality.

Although it provides significant improvements compared to current techniques, the VA-file itself suffers from various problems. In this paper, we first discuss these problems and propose a solution that renders the vector approximation idea more effectively. The major problems of VA-file are the assumption of independent or uncorrelated dimensions, the uniform bit allocation, and the simple partitioning technique. Real data sets are not uniformly distributed, the dimensions of the feature vectors usually are dependent or correlated. More careful analysis for non-uniform and correlated data is needed for effective high dimensional indexing. We propose VA⁺-file, a high dimensional indexing technique which is much more suitable for non-uniform data sets. Our technique can be applied to improve the performance of recently proposed quantization based approaches such as IQ-tree [17] and A-tree [18]. We focus on performance of NN queries, however our analysis can be easily extended to others such as point and range queries.

In typical applications, the amount of data is very large and focusing on exact results may lead to significant inefficiencies in the system. For this purpose, we develop a general framework for approximate nearest neighbor queries, and show that significant speedups can be achieved by sacrificing a little accuracy. Current approaches can be categorized based on either their ability to reduce the data set that needs to be examined, or their ability to reduce the information of each data object. We examine VA⁺-file under one of these categories, and present a simple way to extend it for approximate NN queries. Approximate query processing is more meaningful if the search is performed in a progressive manner. The user can be satisfied with an approximate answer early in the search, or an incomplete result stating that query is leading to an uninteresting data set. We first propose modifications to well-known techniques to support the progressive processing of approximate NN queries. This can be quite beneficial since obtaining exact results may take a long time, and in fact may be unnecessary. We then develop a new technique based on clustering that merges the benefits of the two general classes of approaches. Our cluster-based approach allows

a user to progressively retrieve the approximate results with increasing accuracy.

Section 2 describes the indexing technique based on vector approximations and discusses the importance of approximation quality in such a technique. In Section 3, we motivate the need for a technique for non-uniform data sets and propose the VA⁺-file, a new technique that involves a careful analysis for non-uniform or correlated data. We use several illustrative examples in Section 4 to discuss the impact of better approximations to the efficiency. Second part of the paper focuses more on the effect of allowing approximations in NN searching. Section 5 discusses the notion of approximate searching, introduces the main metrics for evaluating approximate NN queries, and categorizes the current approaches. In Section 6, we develop simple progressive approximation techniques. We then propose in Section 7 a cluster-based integrated approach, and perform in Section 8 an extensive performance evaluation of proposed techniques and show that they result in significant improvements over the state-of-the art techniques. Section 9 concludes the paper with a discussion.

2. Vector approximation based indexing

The VA-file approach divides the data space into 2^b rectangular cells where b is the total number of bits specified by the user [16]. Each dimension is allocated a number of bits, which are used to divide it into equal populated cells on that dimension. Each cell has a bit representation of length b which approximates the data points that fall into a cell by the corresponding bit representation of the cell. The VA-file itself is simply an array of these bit vector approximations based on the quantization of the original feature vectors. An example of a regular VA-file partition of the data space is given in Fig. 1. The two-dimensional data set shown is actually created by taking two dimensions of feature vectors of real time-series data which contains stock price movements of 2000 companies. For simplicity, each dimension is assigned 2 bits, and hence each dimension is divided into 4 intervals of equal population.

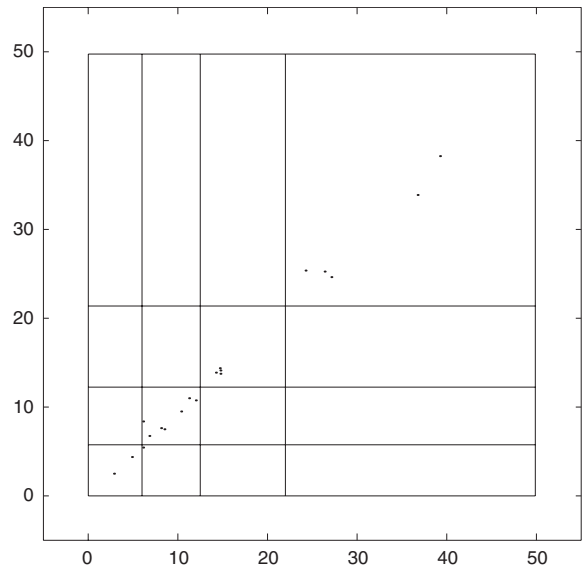


Fig. 1. Regular VA-file partition of the two-dimensional data set. Each dimension is divided into 4 equally populated intervals.

NN searching in a VA-file has two major phases [16]. In the first phase, the vector approximations are scanned sequentially and lower and upper bounds on the distance of each vector to the query vector are computed. If a lower bound exceeds the (k th) smallest upper bound found so far, the corresponding data object is eliminated. In the second phase, the original data objects of the candidates are traversed in the order of the lower bounds. If a lower bound is greater than the actual distance of (k th) NN so far, then the algorithm stops retrieval. The accesses in the second phase are mostly secondary storage accesses. It is crucial to decrease the number of vectors visited in the second phase to reduce the number of random I/O .

The performance of the VA-file approach heavily depends on the quality of the lower and upper bounds. In particular, tighter lower bounds result in earlier termination of the second phase, and hence less number of page accesses. Together with tight upper bounds, better lower bounds also mean less number of candidates remaining for the second phase. If the lower and upper bounds on the distances are tight, more elimination is performed in the first phase of the algorithm. A

more proper index design will clearly maximize the data pruned and greatly reduce time to answer the query.

3. VA⁺-file: indexing high dimensional non-uniform data

In the regular VA-file approach, there is an assumption that the dimensions are independent, or at least uncorrelated, because each dimension is divided into cells independently. Also, although there is always an option of non-uniform bit allocation among dimensions, no specific algorithm for that option was proposed. Finally, each dimension i is divided into 2^{b_i} cells of either equal size, or equal population, which are the two simplest partitionings that coincide for a uniformly distributed data. In this section, we mention the major problems that need to be overcome and propose approaches that lead to more efficient searching using vector approximation files. In particular, we highlight some of the problems that VA-files suffer if the data set is not uniformly distributed, especially when it is highly correlated or clustered. We then describe the VA⁺-file which is much more suitable for such data sets.

3.1. Motivation

A VA-file can be considered as a scalar quantizer [19] without representative values assigned to each cell. The objective of an effective scalar quantizer is to achieve the least reproduction error, i.e., the least average Euclidean distance between data points and their representatives. We claim that a scalar quantization designed by directly aiming for the least possible reproduction error would result in much tighter lower and upper bounds for the distances between the query point and the data points. As discussed in the previous section, tighter lower bounds mean less number of vectors visited in the second phase of the VA-file algorithm, and tighter upper bounds mean better filtering of data in the first phase.

A better approach for designing a scalar quantizer should follow these steps [19]:

1. The data should be transformed into a more suitable domain.
2. In general, the total bits in the quota should be allocated among the transformed dimensions non-uniformly.
3. An optimal scalar quantizer should be designed for each transformed dimension independently, with the allocated number of bits. In general, the quantizers should not assume uniform data and should make use of the data statistics.

3.2. VA-file in KLT domain

It is a well known fact that better scalar quantization performance is achieved when the dimensions of the data are uncorrelated. So, if the dimensions are highly correlated as in the example shown in Fig. 1, or even slightly correlated, it is in general beneficial to decorrelate them by applying a unitary transform to the data set. A transform is called unitary if it preserves all the “angles and lengths” in the data space. If the process generating the data set is known, a fixed transform might be suitable. For example, if we have a time-series data, the discrete fourier transform (DFT), the discrete cosine transform (DCT), or the discrete wavelet transform (DWT) will be effective in decorrelating the data. However, in general, the optimal approach is to estimate the correlation matrix for the given data set, and derive the unique decorrelating transform for it. The resultant transform is known as the Karhunen Loeve transform (KLT) [20,21].

Basically, the function of KLT is to strip off all the correlation between the dimensions of feature vectors. That task is achieved by the diagonalization of the *autocovariance matrix* \mathbf{C} , whose entries are given by

$$C_{ij} = E\{(X_i - \mu_i)(X_j - \mu_j)\},$$

where $E\{\cdot\}$ denotes the expectation operator, X_i is the random variable governing the i th data dimension, and

$$\mu_i = E\{X_i\}.$$

In vector notation, this implies

$$\mathbf{C} = E\{(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^T\}.$$

In other words, the covariance matrix is computed by first subtracting the center of mass of the whole feature vector distribution from each individual vector, and then computing the cross-correlation between all dimensions. In practice, one does not have direct access to the distribution of \mathbf{X} . However, if we have a large database, then the observed feature vectors are considered to accurately model \mathbf{X} . In particular, we can use the approximations

$$\boldsymbol{\mu}' = \sum_{n=1}^N \mathbf{x}(n)$$

and

$$\mathbf{C}' = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}(n) - \boldsymbol{\mu}')(\mathbf{x}(n) - \boldsymbol{\mu}')^T,$$

where $\mathbf{x}(n)$ denotes the n th feature vector, and N is the number of entries in the database. The transformed vectors are then computed as

$$\mathbf{t}(n) = \mathbf{K}(\mathbf{x}(n) - \boldsymbol{\mu}'),$$

where \mathbf{K} , the KLT-matrix, satisfies

$$\mathbf{K}\mathbf{C}'\mathbf{K}^T = \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{bmatrix}$$

with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$. In other words, rows of \mathbf{K} are the *eigenvectors* of \mathbf{C}' , and λ_i are the corresponding *eigenvalues*. One of the important properties of KLT is that λ_i gives the variance (or energy) of the i th transform dimension. Therefore, the ordering of the eigenvalues becomes crucial in guaranteeing that keeping the first few dimensions of the transform results in minimal energy loss in the representation of the feature vectors.

For the example given in Fig. 1, KLT will basically result in a 45° rotation, because the dimensions are almost maximally correlated, i.e., the data is almost spread through a straight line with slope 1. In Fig. 2, we present the resultant VA-file after transforming the data set using KLT.

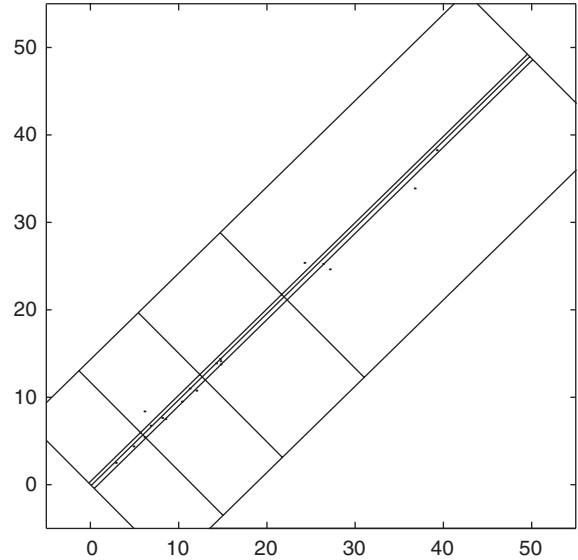


Fig. 2. Partitioning of the same data set after applying KLT. Each KLT-dimension is divided into 4 equally populated intervals.

For the sake of being able to compare the result with Fig. 1, the data set and also the VA-file are both rotated back using the inverse KLT. While we develop our framework using KLT, the other dimensionality reduction techniques can be similarly employed.

The VA-file approach depends on quantization performed on each dimension independently. When there is correlation between dimensions, independent quantization on dimensions does not capture some important information about the data point distribution. On the other hand, KLT rotates the data set so that there is as little dependency between the dimensions as possible. This phenomenon can be understood by comparing Figs. 1 and 2. The VA-file creates cells in non-transformed domain based on having equal population on the x and y dimensions (Fig. 1). This partitioning is optimal only if the data set is uniformly distributed. In the example shown in Fig. 1, several boxes have similar square-like area and the cells around the diagonal of the data space overflow, while in Fig. 2 the data objects are more equally distributed among the cells. The advantage of KLT becomes much more apparent when it is used as a preprocessing step for a more accurate

analysis of the data set. The KLT transforms the data set into a more suitable domain where the subsequent steps of our algorithm, i.e., non-uniform bit allocation and independent quantization for each dimension, improve the performance by taking advantage of the statistical properties of the data set.

3.3. Non-uniform bit allocation

It is obvious from Fig. 2 that allocating the bits in our quota, i.e., the total number of bits that is allowed for approximating the data vectors, non-uniformly among the KLT-dimensions would result in better quantization performance. For example, if we allocate 4 bits to the major axis of the data, and no bits to the minor axis, thereby keeping the total as 4 bits, we obtain the vector approximation file shown in Fig. 4. The information that is stored in each dimension of the feature vector varies depending on the feature vectors. The goal is to approximate this information with a minimum number of bits with maximum accuracy. Therefore, for several cases it is crucial to analyze the dimensions and allocate the bits to the dimensions, rather than the simplistic uniform bit allocation, so that the resulting accuracy obtained from the approximation is maximized.

Non-uniform bit allocation is an effective way to increase the accuracy of the approximations for any data set. However, it becomes more advantageous and more feasible in the KLT domain. The difference between the amount of energy stored in each dimension becomes much more prominent when dimensions are transformed using KLT. KLT is known to have maximum energy compaction property for any given data set. That means KLT successfully accumulates the data energy more into the first dimensions.

For the example data set (Figs. 2 and 4), it can be seen that in the KLT domain the y dimensions of the points are already populated in the middle and close to each other, while the x dimensions of the data points are spread throughout the axis. To differentiate and therefore better approximate the x dimension of the points, non-uniform bit allocation allocates more bits to this axis while the total bit quota is kept same as before. With

1. INITIALIZE $e_i = \sigma_i^2$, $b_i = 0$ for all i , and $k = 0$.
2. WHILE $k < b$, DO
3. $j = \arg \max_i e_i$
4. % In other words, $e_j = \max_i e_i$
5. $b_j \leftarrow b_j + 1$
6. $e_j \leftarrow e_j/4$
7. $k \leftarrow k + 1$
8. END

Fig. 3. Non-uniform bit allocation algorithm.

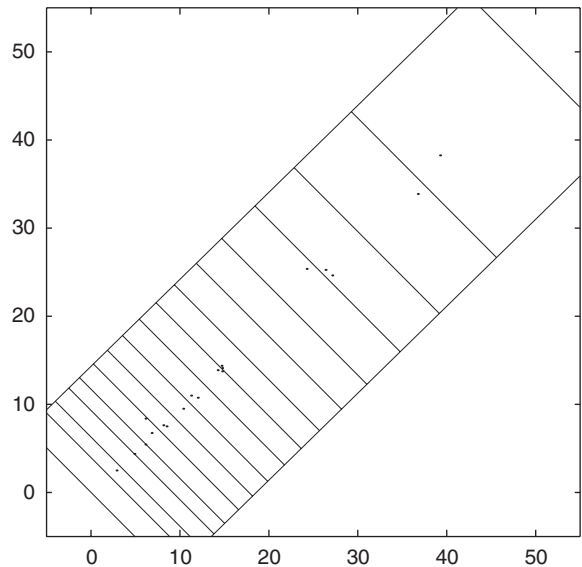


Fig. 4. Partitioning of the same data set after changing the bit allocation. Now, the major axis has 16 intervals, and the minor axis has only 1 interval.

uniform bit allocation, the approximation quality of the cells would decrease since points that are far away from each other are approximated with the same cell.

Let the variance of dimension i be σ_i^2 , and the number of bits allocated to dimension i be b_i . We have a quota of b bits, i.e., $b = \sum_i b_i$. In quantization theory, a well-known rule is that if $\sigma_i^2 \geq 4^k \sigma_j^2$, then it is more beneficial to have $b_i \geq b_j + k$ [19]. In fact, this rule is based on the assumption of high resolution uniform quantization, i.e., equal-sized intervals and $b_i, b_j \gg 1$. Nevertheless, even when we do not have high resolution, and we do not perform uniform quantization, this rule still proves to be a good heuristic. The resultant bit allocation

algorithm for VA-file is given in Fig. 3. Fig. 4 is the result of applying the above bit allocation algorithm to our sample data for a total quota of 4 bits.

3.4. Non-uniform quantization

Once the number of bits are determined for a particular KLT-dimension, it is more advantageous to design a quantizer for it, rather than dividing it into equally populated or equal-sized intervals. A quantizer is characterized by the decision intervals, and the corresponding representative values for each interval. Even though the representative values are not going to be used directly, they also need to be designed together with the decision intervals. A very popular algorithm for designing a quantizer with K intervals is Lloyd's algorithm [22,23], which is also known as the K -means algorithm in the clustering literature [24], specialized to one dimension. The algorithm is initialized with a given set of intervals $[c_j, c_{j+1})$ for $j = 1, \dots, K$, where $c_1 = \min_n t(n)$ and $c_{K+1} = \varepsilon + \max_n t(n)$. See Fig. 5 for the description of the algorithm in pseudocode. Since $0 \leq \Delta' \leq \Delta$ provably, this algorithm is guaranteed to converge. However, it may get trapped into poor local minima, and hence needs clever initialization. For example, to initialize the algorithm with the equally-populated cells, i.e., the current VA-file approach, is a powerful heuristic. Following this initialization technique yields a resultant vector approximation file, i.e., VA⁺-file, as shown in Fig. 6. After applying this third step, the error introduced by approximating the data points by the corresponding cells is further reduced. Since the representation quality of the cells directly effects the quality of the vector approximation file, this leads to better pruning both in the first and the second phase of searching. The improvements by the VA⁺-file is much more apparent for higher dimensional data sets than the two-dimensional example illustrated in this section. These results are also supported later in the paper by the performance analysis of each technique.

1. INITIALIZE $\Delta = \infty$
2. FOR $j = 1, \dots, K$,
3. $r_j = \frac{1}{N_j} \sum_{t(n) \in [c_j, c_{j+1})} t(n)$
4. % r_j is the representative for interval $[c_j, c_{j+1})$
5. % N_j is the total number of data points in the same interval
6. FOR $j = 2, \dots, K$,
7. $c_j = \frac{r_{j-1} + r_j}{2}$
8. $\Delta' = \sum_{j=1}^K \sum_{t(n) \in [c_j, c_{j+1})} [t(n) - r_j]^2$
9. IF $\frac{\Delta'}{\Delta} < \gamma$,
10. % γ is a preset threshold about 0.999
11. $\Delta \leftarrow \Delta'$
12. GO TO 2
13. ELSE STOP

Fig. 5. Non-uniform scalar quantization algorithm.

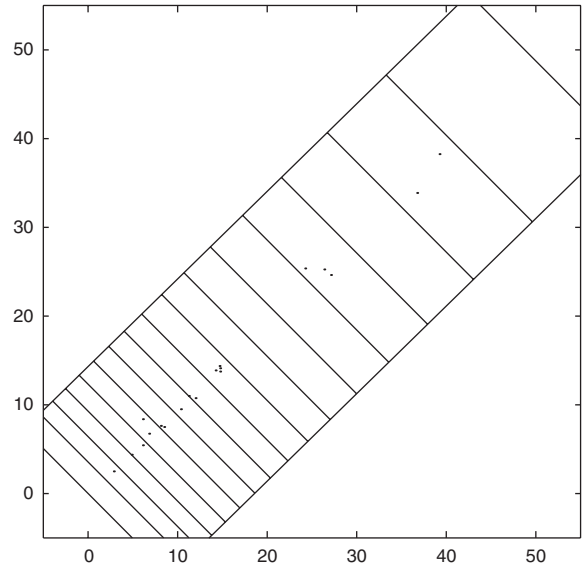


Fig. 6. Partitioning of the same data set after applying Lloyd's algorithm.

4. Illustrative example on importance of approximation quality

In this section, we illustrate the impact of tighter bounds as well as the approximation quality improvement achieved by our technique over the VA-file. The detailed performance evaluation of the VA⁺-file is presented later in the paper (see Section 8). To clearly present the importance of

the bounds, the discussions in this section are done over an illustrative query. However, as will be seen in the performance evaluation section, the VA⁺-file keeps its superior performance on the average over the VA-file.

We have used a 100,000 60-dimensional vectors representing texture feature vectors of Landsat images [25]. The feature vectors are texture information of blocks of large aerial photographs computed using Gabor filtering. This data set is widely used by many researchers in evaluating high dimensional indexing and similarity searching techniques [26–28]. For this section, we assumed a total bit quota of 180 bits, i.e., 3 bits per dimension.

In Fig. 7, the tightness of the lower bounds achieved by the VA-file and the VA⁺-file are compared using an example. In both cases, lower bounds are calculated for the same typical query point, and the data is re-sorted in ascending order of lower bounds, because this is exactly the order of visiting of the vectors in the second phase. The *x*-axis represents the data objects in resorted order and the *y*-axis represents the distances to the given query point. The dots in the figure correspond to the *actual* distances between the re-sorted data points and the query point. Since the sorting is done according to the lower bounds, the plot for the corresponding lower bounds for each data point form a nice curve, instead of scattered dots,

below the actual distances. We compare the number of vectors visited for *k*-NN queries, where *k* = 10, *k* = 50, and *k* = 250. The algorithm has to visit all vectors with lower bounds less than the distance of the *k*th nearest data. The intercepts of dashed lines on the *y*- and *x*-axes, respectively correspond to the distance of the *k*th nearest data, and the number of vectors visited. As shown in the figure, as the lower bounds get tighter, the data becomes more suitably sorted, i.e., more or less in an ascending order of real distances to the query point. Hence, the actual *k* neighbors are found by visiting less number of vectors (= accessing less number of pages). For example, for this query point with *k* = 10, the *k*th nearest neighbor has a distance around 62, and the VA-file method has to visit 15 times as many vectors as the VA⁺-file method does.

Fig. 8 shows that tighter lower bounds together with tighter upper bounds imply better filtering in the first phase of the *k*-NN algorithm. To illustrate this important property, the same query point as before is chosen. Cumulative distributions of real distances, lower bounds, and upper bounds are shown together for both the VA-file and the VA⁺-file methods. A cumulative distribution is defined as the integral (or cumulative sum) of a regular probability density function, and is more appropriate to consider for our purposes here. For example, if the lower bound cumulative

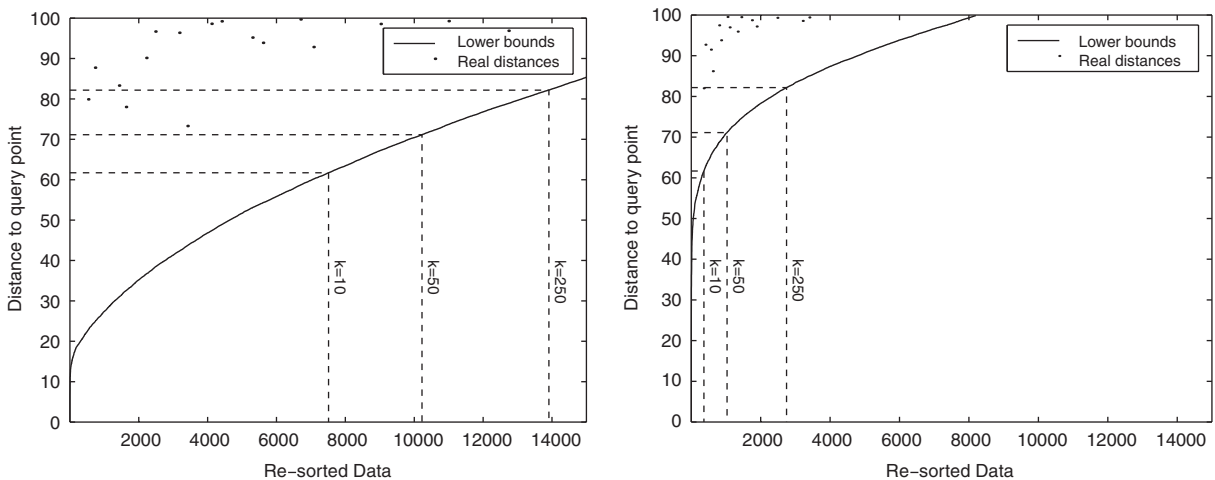


Fig. 7. Comparison between VA-file (left) and VA⁺-file (right), in terms of tightness of lower bounds and number of vectors visited.

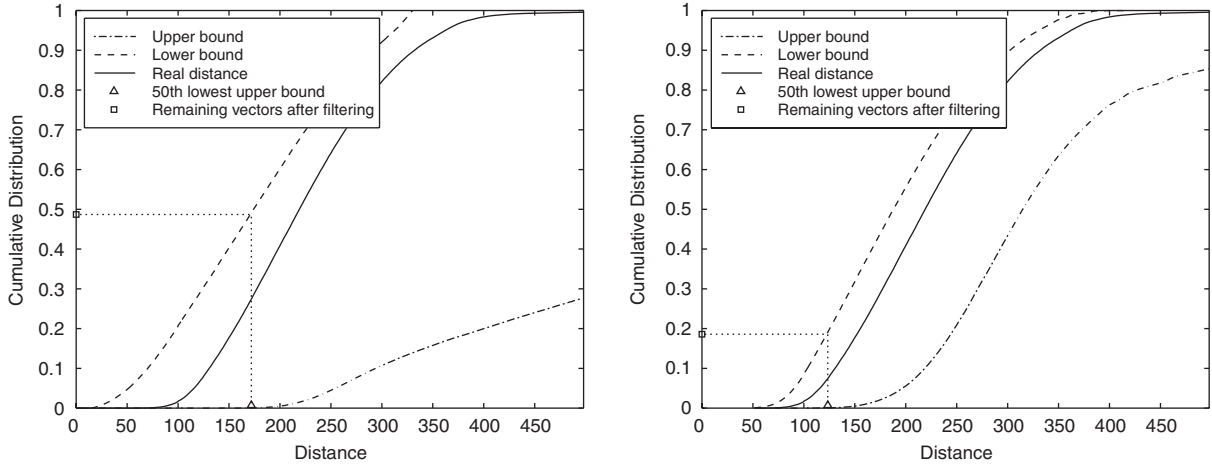


Fig. 8. Comparison between VA-file (left) and VA⁺-file (right), in terms of the cumulative distributions of lower and upper bounds, and number of vectors eliminated in the first phase.

distribution takes the value 0.2 for distance 25, it means that one fifth of the data has lower bounds less than or equal to 25.

For a k -NN query, all vectors having lower bounds less than the k th lowest upper bound remain unfiltered in the first phase. In fact, if the data were sorted in the best possible order, only those vectors would have remained after the filtering. In the figures, k is set to 50, and the triangle on the x -axis shows the value of the 50th lowest upper bound, and the square on the y -axis shows the fraction of vectors that remained unfiltered for this best case scenario. Our experiments show that the actual fraction of unfiltered vectors are very close to the best-case value and not plotted here for the sake of clarity. Tighter upper bounds reduce the value of the 50th lowest upper bound, and hence reduce the fraction of the vectors that remain unfiltered after the first phase. On the other hand, tighter lower bounds also reduce the same fraction, because less vectors would have lower bounds less than the 50th lowest upper bound. This phenomenon is clearly observed from Fig. 8, where the VA⁺-file displayed on the right has much tighter lower and upper bounds for the fixed query point than the VA-file displayed on the left.

In Fig. 9, the cumulative distributions of real distances, lower bounds, and upper bounds of

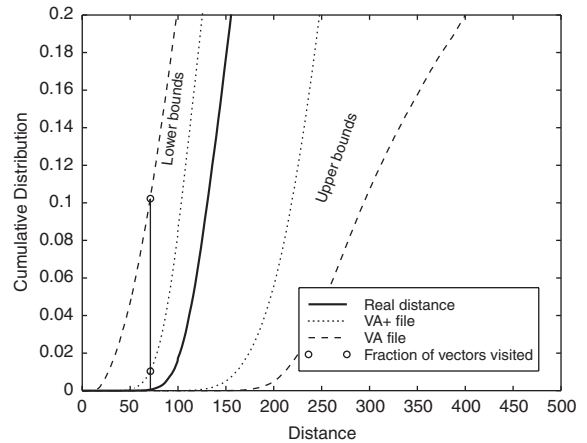


Fig. 9. Comparison between VA-file and VA⁺-file, in terms of the cumulative distributions of lower and upper bounds, and number of vectors visited in the second phase.

both methods are shown on the same plot. Also shown by circles are the fraction of vectors visited in the second phase, again for both algorithms. The fraction of vectors visited corresponds to the value the lower bound cumulative distribution attains for the 50th lowest distance. As seen from the figure, the VA-file method visits around $100,000 \times 0.09 = 9000$ vectors, whereas the VA⁺-file visits only $100,000 \times 0.015 = 1500$ vectors.

5. Approximate NN searching

The focus of VA⁺-file has been optimizing exact NN searching. We now extend our discussion to approximate searching. In some applications, the amount of data is very large and focusing on exact results may lead to inefficiencies in the system. In these cases, achieving significant performance improvements by sacrificing little accuracy can be especially attractive. The generation of feature vectors in several modern database applications are based on heuristics. Besides the semantics of the data, the semantics of queries are also not as strict as the exact queries used in relational databases. For example, a query asking the closest 5 Italian restaurant to a moving car may easily miss the 3rd closest and include the 6th closest, and still satisfy the driver. The definition of ‘exact’ is subjective and depends on the way the feature vectors are created and the distance function defined between the feature vectors. For example, the QBIC project at IBM provides the ability to run queries based on colors, shapes, and sketches [1,29]. Similarly, the Alexandria Project at UC Santa Barbara provides similarity queries for texture data [25]. One user may choose the third best result over the first or the second one, based on his/her similarity expectation. In several cases, close approximations may be good enough for human perception. In this section, we develop a general framework for approximate nearest neighbor queries. We provide performance metrics, and classify current approaches, including VA⁺-file, under two major classes. Since approximate query processing gains more meaning with being progressive, in the following sections we discuss extensions of well-known methods for progressive searching. We then develop a new technique based on clustering which allows a user to progressively retrieve the approximate results with increasing accuracy.

5.1. Approximation quality metrics

For similarity queries, the quality of the result set is traditionally measured by a combination of two important quality metrics: *recall* and *precision* [30]. They can be described as completeness of

retrieval and purity of retrieval, respectively. Recall is a measure of how well the retrieval finds all relevant objects even to the extent that it includes some irrelevant objects. Precision is a measure of how well such a system finds only relevant objects even to the extent that it skips some relevant objects. The irrelevant objects in the result set are called *false hits* and the relevant objects that are not in the result set are *false dismissals*. A traditional approach which uses these two metrics is dimensionality reduction on multi-dimensional data. In general, high dimensional data is first reduced to lower dimensions and the search is conducted in the lower dimensional space [31]. Most of the dimensionality reduction techniques focus on allowing some false hits but no false dismissals.

For approximate k -NN searching, the number of false hits and the number of false dismissals become the same value. Since the result set size is always k , if the approximation causes some false dismissals, these dismissals are replaced by false hits. Computing the number of false hits, or dismissals, is enough to capture the traditional error metric, which we will refer to as F . We use the metric F as one of our metrics in the evaluation of the proposed techniques.

However, the number of false hits and dismissals does not capture important information about the quality of the approximations. In Fig. 10(a), the two NN of the query point q are asked. The points a and b are the two closest points according to the Euclidean distance. However, the points c and d are also very close to the query point and the user may potentially be interested or satisfied with them being the answer. On the other hand, the points e and f are far away from q and there is much less chance that the user is interested in those points. Suppose there are two approximation techniques, one returns (c, d) and the other returns (e, f) as the result of 2-NN query on q . If the traditional error metric of the number of false hits is used, both techniques have the same number of false hits, i.e., 2, which is the worst possible error. Therefore, with this metric both techniques have the same error and it is not possible to differentiate the qualities of these two different answer sets.

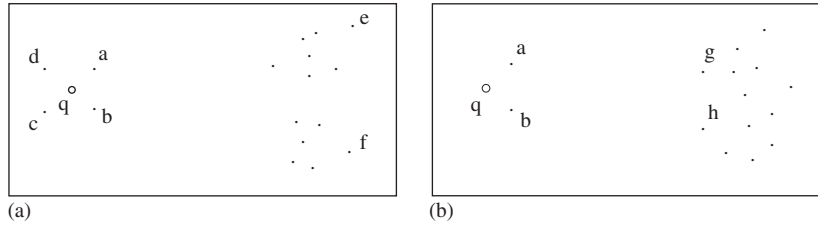


Fig. 10. 2-NN query on q : (a) Example 1; (b) Example 2.

The points c and d in Fig. 10(a) are not only the third and the fourth closest points to the query, they are also spatially close enough to be of interest to the user. It is also important to note that the rankings of the points in terms of closeness to the query does not capture any information regarding the quality of approximation. For example, points g and h in Fig. 10(b) are not of that much interest even though in this case, they are also the third and the fourth closest points to q . The quality/error metric should give a higher error to (g, h) in Fig. 10(b) than it gives to (c, d) in Fig. 10(a). Especially when dealing with approximate similarity-based retrieval, it is important to develop a new metric which also takes into account the quality of the answers with respect to closeness to the query object.

We now introduce an alternative error metric which is particularly useful for approximate k -NN searching. Suppose the approximate k -NN algorithm returns the result set (a_1, a_2, \dots, a_k) and the real result set computed by the underlying distance function d_f is (r_1, r_2, \dots, r_k) . We define the error metric D as the relative approximation error, given by $\sum_{j=1}^k d_f(q, a_j) / \sum_{j=1}^k d_f(q, r_j)$. For instance, for squared Euclidean distance D is defined as follows:

$$D = \frac{\sum_{j=1}^k \|q - a_j\|^2}{\sum_{j=1}^k \|q - r_j\|^2}.$$

5.2. Approaches for approximate searching

We now classify the general approaches that can be used to solve the approximate searching problem. In the following section we will develop various techniques as examples of these general classes.

Consider a data set with n data objects each represented by a d dimensional feature vector. In typical applications, both the dimensionality d and the number of data objects n are large. Problems such as the curse of dimensionality in high dimensions and scalability for very large data sets can partially be offset by structures that support approximate searching. An approximate searching technique should maximize the accuracy and minimize the cost of processing queries, which are usually dominated by the number of I/O operations. Thus, an effective approximate searching technique should organize the data such that acceptable accuracy is achieved and at the same time the number of pages retrieved as a result of a query is minimized.

Naively, a NN query could be answered by examining the *entire* representative of *every* data object. Two general classes of possible approaches for the approximate searching problem naturally arise:

- Retrieved set reduction: the data is organized such that only a small subset of the data objects are examined to answer a NN query.
- Retrieved information reduction: the data is organized such that only a partial representation (e.g., 2 out of d dimensions) of each object is examined.

The *retrieved set reduction* approach is important especially for scalability in large data sets. An approximate NN search technique which retrieves the feature vectors related to a subset of the data set clearly outperforms a technique that retrieves the feature vectors for the entire data set. Similarly, the *retrieved information reduction*

approach is crucial for high dimensional data. Considering all the dimensions at the same time dramatically degrades the efficiency of high dimensional query processing. We briefly discuss some possible techniques that are based on these two basic ideas.

Retrieved set reduction: Instead of accessing/retrieving the entire data set, the search technique focuses on a portion of the data set, ideally on the most relevant data to the given query. One effective and well-known solution is to index the multi-dimensional data. The goal of all these index structures is to reduce the size of the data set which is retrieved as a result of a query. Instead of n objects, the query retrieves a set of s objects, which is a subset of these n objects. The better the index structure, the smaller the size of s . There have been several approaches to organize and partition a multi-dimensional data set for indexing including kdb-trees [9], hB-tree [12], R-tree [10], R*-tree [11], SS-tree [13], TV-tree [32], X-tree [33], pyramid technique [34], hybrid tree [14]. These index structures focus mainly on finding the exact result for a query. Therefore, a retrieved set of size s includes the query result set and some irrelevant objects. The optimal set of objects to be retrieved is the result set, i.e., no objects are retrieved if they are not in the result set. A branch-and-bound technique for k -NN queries on index tree structures, such as R-tree, is proposed in [35]. The tree, which consists of minimum bounding rectangles (MBR), is traversed and some of the MBRs are pruned if they are guaranteed not to have the closest data point(s). For example, if the shortest possible distance between the query point and any point in an MBR is larger than the current computed NN distance, then there is no need to check the data points within that MBR. The amount of data retrieved as the result of a query is reduced by pruning some of the MBRs using proximity comparisons of points within MBRs.

In the context of approximate searching mechanism a natural question arises: is it possible to achieve faster response time if we allow some false dismissals? Recently, various approaches in different domains have developed effective algorithms for approximate searching [27,28,36–41]. Most of these techniques specifically focus on ε -NN

queries. The ε -NN is defined as finding a neighbor of the query point within a factor of $(1 + \varepsilon)$ of the distance to the true NN. In [28], an algorithm was proposed in which the error bound ε can be exceeded with a certain probability δ using some prior information on the distance distribution of the query point. This technique can be classified as a retrieved set reduction approach, which reduces the number of retrieved data by pruning a larger number of index nodes. The retrieved data set size is first reduced by the underlying index structure. Besides the irrelevant objects pruned by the index, more objects are pruned by shrinking the NN query sphere at the expense of allowing some false dismissals. The technique also adds a second level of approximation, where more objects can be pruned by allowing to exceed ε with the probability δ . Trading space with time using retrieved set reduction is discussed also in [42] where a structure called P-sphere tree (probabilistic sphere tree) is proposed for processing NN queries. In [27], a locality sensitive hashing structure is created by a randomized procedure and the $(1 + \varepsilon)$ -approximate NN point is found with a constant probability. Hashing is used to reduce the retrieved set size, therefore improving query time, again at the expense of false dismissals. This approach is also based on the retrieved set reduction idea, where hashing is used instead of a multi-dimensional index tree to identify the retrieved data set. The disadvantage of this approach, however, is that ε needs to be known in advance, and some preprocessing, which is exponential in $1/\varepsilon$, is needed.

Retrieved information reduction: Multi-dimensional index trees are very effective in low dimensions. Therefore, they are widely used in low dimensional applications such as GIS [3]. However, there are several applications, e.g., multimedia databases [1,43], that need high dimensional support. It is well known that as the dimensionality of the feature vectors increases, query performance of the multi-dimensional index tree structures degrades significantly [15,16,33,44]. Therefore, considering all the dimensions at the same time dramatically degrades the efficiency of high dimensional query processing. Retrieved information reduction approach is particularly

effective to this problem. A typical example to this approach is the dimensionality reduction. The most common approaches found in the literature for dimensionality reduction are linear-algebraic methods such as the KLT [20,21], or applications of mathematical transforms such as the DFT [45], DCT [46], or DWT [47]. As the transformations are known to be distance preserving, the general approach is to transform the high dimensional feature vectors and lower dimensional vectors are created by taking the first few leading coefficients of the transformed vectors [31]. The general idea of these techniques depends on the observation that by using these transformations, a small subset of dimensions keeps a high portion of the information about the feature vectors. Several other dimensionality reduction based techniques are proposed [34,48,49] and applied to time-series [50,51], image [29], and document data [52–54]. For dynamic data sets, approximate KLT has been shown to be an effective technique compared to the techniques based on exact transformations [55].

VA-file and VA⁺-file are effective examples of retrieved information reduction which can be used for high dimensional exact NN searching. In both approaches, the total size of the feature-vector set is reduced by a significant amount using quantization of the original feature vectors. As discussed earlier, exact NN searching in a VA-file has two major steps. In the first step, the set of all vector approximations is scanned sequentially and lower and upper bounds on the distance of each vector to the query vector are computed. The vectors with the smallest bounds are determined to be the candidates for the NN of the query. Subsequently, in the second step the real feature vectors of the candidates are visited to determine the actual NN. Recently, Weber and Bohm [38] have stated that the performance of previous approaches for approximate NN-search, with reasonable approximation errors, is close to linear. They propose an approximate k -NN searching technique based on VA-files. Two approaches for approximate searching with comparable performances are proposed in the paper [38]. One approach is to create coarser approximations based on a pre-analysis of the data allowing false drops. This approach is more useful

if one wants to accelerate the computations in the first step. To overcome the *I/O* bottleneck which is important for large databases, they proposed a VA-file based technique which omits the second step of the exact NN search algorithm. The similarity distances are estimated from the lower and upper bounds in the first step, and the result set is created using the results in the first step. By allowing some errors in the result set, approximate NN searching in VA-file achieves an order of a magnitude speedups over exact NN search in VA-file. The same extensions can be applied to VA⁺-file. In the performance evaluation section, we also implement the approximate NN algorithm for the VA⁺-file and compare the performance behavior with other approaches.

Several techniques can be developed that covers both retrieved set reduction and retrieved information reduction. For example, using a dimensionality reduction technique before building an index on the data simply combines the two approaches. A more effective approach would be to integrate the two classes under a single framework, as opposed to separately applying them. We will introduce such a technique in Section 7.

6. Simple progressive approximate search

We refer to progressive approximation as an interactive technique, where the user is able to stop the query anytime he/she thinks it is appropriate and still gets a result with a certain error. It is important to adapt the approximation quality using the user as the reference point. If the user wants to get more accurate results, he/she can keep the query running until the desired result is obtained or no more data is available. Without waiting for the end of the query, which typically takes a long time, the user may also want to stop the search once he/she understands that the data set most likely does not have the desired answers. For each step of an interactive search, the expected quality can be estimated by a pre-analysis of the data set, as is done in this section.

In this section, we develop two simple approximate k -NN techniques that are instances of retrieved set reduction and retrieved information

reduction approaches. These approaches modify the well-known techniques and adapt them to support the progressive retrieval of approximate information based on NN queries.

6.1. Sequential scan with retrieved set reduction

An obvious and simple way of implementing the retrieved set reduction is to sequentially scan a portion of the data set. The basic idea is to access only a portion of the data set and answer the query based on the portion that is read. Since sequential scan is known to be an effective alternative to several indexing techniques for high dimensional data sets [15,16,44], it is natural to start the discussion of approximating k -NN queries with sequential scan. This simple idea can be summarized as follows. The data set is stored in the storage without any particular index structure. When a query is issued, the data is scanned sequentially starting from the first page. The data pages are read in the order they are stored and k -NN can be computed within the portion of the data set that is read so far. The search can be interactive, and it can be stopped any time, and the answer is based on the portion of the retrieved data set.

6.2. Retrieved information reduction: sub-vectors based on KLT

We now illustrate how distance preserving transformation and dimensionality reduction techniques can be adapted for progressive approximate k -NN searching. Distance preserving transformations, e.g., KLT, are especially suitable for interactive approximate searching. However, instead of retrieving each of the feature vectors at once with all its dimensions, a better strategy is to split the feature vectors into blocks and perform multiple passes on each feature vector by retrieving the blocks in the order of importance. A similar approach has been followed by Vries et al. [56] in the context of vertically decomposed data. This helps to gather more useful information, i.e., the important dimensions of each feature vector, in earlier steps and gather less important information later in the search, if necessary. Instead of waiting for a whole pass on all feature vectors, the

system can provide the user an accurate approximate result that is obtained from the first set of dimensions which have a high amount of energy.

Our approach for feature-vector set organization based on distance preserving transformations works as follows. The d -dimensional feature vectors, where d is usually high in typical applications, are transformed using the KLT and a new set of d -dimensional vectors is created. This set is used as the new set of feature vectors. We use KLT, because it is known to have the maximum energy compaction property for any given data set. That means, among all the transform-based dimensionality reduction techniques, KLT is the best in accumulating the data energy into a fixed number of dimensions. Each feature vector is divided into s fixed sub-vectors, each with r_i number of dimensions, where $\sum_{i=1}^s r_i = d$. The corresponding sub-vectors of all feature vectors are stored consecutively and therefore the sub-vectors of each feature vector are stored separately. Without loss of generality let $r_i = 1$, for $0 \leq i \leq d$, and therefore $s = d$ for simplicity. In this case, the values of first dimension of each vector are stored together consecutively on secondary storage, and second dimension values are stored together, and so on. See Fig. 11 for the algorithmic description of this setup phase.

When a query is issued, in the first step, only the pages that contain the first dimensions of all vectors are read from storage and the first dimensions are considered for similarity. Then, the other dimensions are read and the similarity computation is updated based on all dimensions that have been read. Partial results are stored in memory to be used for the consecutive steps. If the

1. $\mu' = \sum_{n=1}^N \mathbf{x}(n)$
2. $\mathbf{C}' = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}(n) - \mu')(\mathbf{x}(n) - \mu')^T$
3. COMPUTE \mathbf{K} so that $\mathbf{K}\mathbf{C}'\mathbf{K}^T = \mathbf{\Lambda}$
4. FOR $n = 1, \dots, N$,
5. $\mathbf{t}(n) = \mathbf{K}(\mathbf{x}(n) - \mu')$
6. FOR $i = 1, \dots, d$,
7. FOR $n = 1, \dots, N$,
8. WRITE $t_i(n)$

Fig. 11. Setup phase of the algorithm.

1. GET \mathbf{q} (query) and k
2. $\mathbf{q}' = \mathbf{K}(\mathbf{q} - \boldsymbol{\mu}')$
3. FOR $n = 1, \dots, N$,
4. $dist(n) = 0$
5. FOR $i = 1, \dots, d$,
6. FOR $n = 1, \dots, N$,
7. READ $t_i(n)$
8. $dist(n) \leftarrow dist(n) + [q'_i - t_i(n)]^2$
9. SORT $dist$
10. PRINT indices of smallest k entries of $dist$
11. IF user satisfied, STOP
12. END

Fig. 12. Execution of k -NN search in the KLT domain.

user stops the search anytime, the query is answered based on the results that are found up to that point. Later in the paper, we will evaluate the performance of this simple approach and compare it with the other approaches. Fig. 12 shows the algorithm in pseudocode.

7. An integrated approach for approximate searching

Both retrieved data set and retrieved information reductions have strong motivations and need to be considered. Therefore, it is important to develop techniques that combine the advantages of both retrieved set and retrieved information reductions. We propose a new technique that effectively combines the two class of approaches into a single framework. The retrieved portion of data is reduced by the help of a clustering technique, which is an adaptation of K -means clustering [24,57], and the feature vectors within a cluster are organized to support progressive approximate searching.

First, a dimensionality reduction, from d to r , is performed on each data point in the data set. The dimensionality reduction is done as described earlier, i.e., by taking the first r dimensions in the KLT domain. The value of r can be determined based on a statistical analysis of the data. For all dimensions i , the average energy that is stored in the first i dimensions is computed. Let r be the

minimum number of dimensions that keeps energy that is above a certain threshold. Analysis of the energy stored in KLT-domain dimensions can be used for high dimensional data characterization. The r value that stores an amount of energy that is greater than a certain threshold, say 85%, is a simple characterization. We refer to the r dimensions which store more energy than the threshold, the dominant KLT-dimensionality of the data. In our analysis, we establish that the first few dimensions, usually 5–10 dimensions, in the KLT domain store a high amount of the energy. Fig. 13 illustrates the percentage of cumulative energy accumulated into reduced number of dimensions for three different real data sets, which we will describe and use later in the performance evaluation, i.e., image color histogram, stock time-series, and satellite image textures data sets.

After dimensionality reduction, we employ a modified K -means clustering algorithm in the low dimensional domain. The original K -means algorithm [24,57] iteratively constructs a number of clusters with a representative for each cluster such that the error in representation within each cluster is minimized. The K -means algorithm has found applications in many different disciplines: unsupervised learning in pattern recognition [57], unsupervised image segmentation [58], and vector quantizer design in signal compression [23]. Various extensions of K -means have been proposed in the database literature, e.g., for wavelet-based incremental clustering of time-series data [59].

The modification we introduce in the original K -means algorithm is that we limit the size of each cluster from both above and below. If the size goes above the upper threshold, the cluster is split into two. If the size goes down below the lower threshold, then the cluster centroid is erased from the list of centroids. To split a cluster, we first duplicate the cluster centroid, and then perturb the exact copies randomly. It is known that K -means algorithm is sensitive to initialization. Since we have this splitting mechanism, instead of starting from cluster centroids chosen by some pre-processing scheme, we start by a single cluster, and the algorithm automatically creates new clusters until the population of each cluster is below the

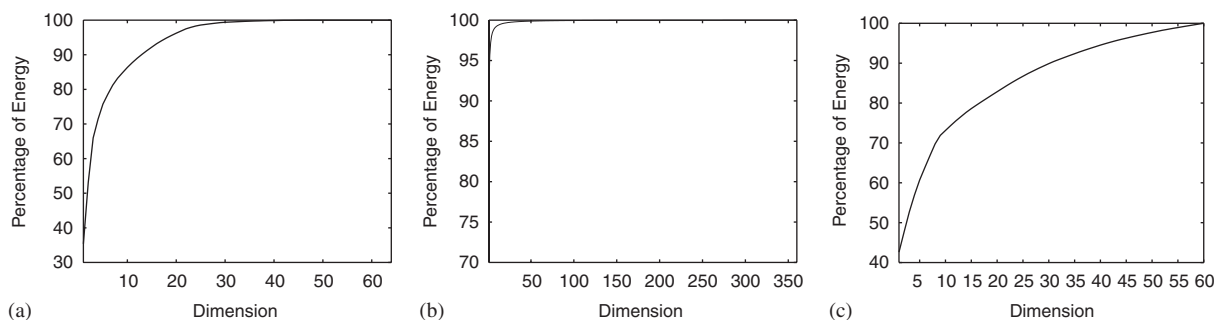


Fig. 13. Percentage of cumulative energy accumulated into reduced number of dimensions: (a) Color Histogram; (b) Stock Market Time-Series; (c) Satellite Image Texture.

```

1. SUBROUTINE K-MEANS( $\mathbf{t}(n)$ ,  $\mathbf{c}_i$ ,  $K$ ,  $\gamma$ )
2.    $\Delta = \infty$ 
3.   FOR  $i = 1, \dots, K$ ,
4.      $N_i \leftarrow 0$ 
5.   FOR  $n = 1, \dots, N$ ,
6.      $f(n) = \arg \min_i \|\mathbf{t}(n) - \mathbf{c}_i\|^2$ 
7.      $N_{f(n)} \leftarrow N_{f(n)} + 1$ 
8.   FOR  $i = 1, \dots, K$ ,
9.      $\mathbf{c}_i = \frac{1}{N_i} \sum_{n: f(n)=i} \mathbf{t}(n)$ 
10.   $\Delta' = \sum_{n=1}^N \|\mathbf{t}(n) - \mathbf{c}_{f(n)}\|^2$ 
11.  IF  $\frac{\Delta'}{\Delta} < \gamma$ ,
12.     $\Delta \leftarrow \Delta'$ 
13.    GO TO 3
14.  ELSE RETURN  $\mathbf{c}_i$ ,  $f(n)$ ,  $N_i$ 

```

Fig. 14. The original K -means routine. Here, $\mathbf{t}(n)$, \mathbf{c}_i , $f(n)$, K , and γ respectively denote the transformed and truncated feature vector, centroid of the i th cluster, the mapping from feature vectors to clusters, the number of clusters, and the stopping threshold.

threshold. As we will demonstrate later, by having a lower threshold for cluster size, several queries can be answered by retrieving only a very small number of clusters. Also, by limiting the cluster sizes from above, we avoid extremely unbalanced distribution of data over the clusters. Figs. 14 and 15 describes the modified K -means algorithm in pseudocode. The algorithm calls the original K -means subroutine, which is shown in Fig. 14 for completeness.

```

1. INITIALIZE  $K = 1$ ,  $\mathbf{c}_1 = \boldsymbol{\mu}'$ 
2. finished = FALSE
3. REPEAT UNTIL finished
4.   [ $\mathbf{c}_i$ ,  $f(n)$ ,  $N_i$ ] = K-MEANS( $\mathbf{t}(n)$ ,  $\mathbf{c}_i$ ,  $K$ , 0.999)
5.   finished = TRUE
6.   FOR  $i = 1, \dots, K$ ,
7.     IF  $N_i < L$ 
8.       ERASE  $\mathbf{c}_i$ 
9.        $K \leftarrow K - 1$ 
10.    finished = FALSE
11.   ELSEIF  $N_i > U$ 
12.      $\mathbf{c}_{K+1} = \mathbf{c}_i + \epsilon$ 
13.      $K \leftarrow K + 1$ 
14.     finished = FALSE
15.   END
16. END
17. END

```

Fig. 15. The modified K -means routine.

Although the minimum and maximum cluster sizes, which we denote by L and U , respectively, are not dominant factors in the performance of our technique, reasonable values need to be set for the design purposes. For k -NN queries, k is expected to be less than a given number, say 500, because of the expectations and readability of a query result. In typical applications, k may be taken to be between 1 and 100. Practically, minimum cluster size can have a value comparable to the expected values of k in the application, e.g., $L = 10$, 50, etc. The maximum cluster size can be picked as a function of the minimum cluster size, e.g., $U = 10L$ or $U = 20L$. One can also pick an

initially huge maximum cluster size U , and decrease it gradually in an outer loop, until the number of clusters reach a reasonable number. We adopted this approach throughout this work.

We note that a clustering based organization can be adapted for exact NN searching as well. By pre-computing additional information about each cluster in design time, one can compute lower and upper bounds of the distance of query to each cluster in query time. A possible way is to represent each cluster by a circle with the mean as the center and the line from the mean to its furthest point as the radius, and define the bounds using these circles. During the search process, clusters that have lower bounds greater than the upper bound of any of the clusters can be eliminated without retrieval. Similarly, once a cluster is retrieved and actual distances are computed, clusters with lower bounds greater than the current NN distance can be eliminated from the search process. However, we note that the biggest advantage of clustering for NN searching is its capability of reaching the most relevant data with high probability in the earliest time, which makes it appropriate especially for approximate searching. A very small number of clusters is typically enough to achieve very high accuracy, while a relatively larger number of clusters would be required to guarantee locating the exact NN. Nonetheless, lower and upper bounds to each cluster can be utilized as a stopping criteria for the approximate search algorithm.

The proposed integrated algorithm for approximate k -NN searching works as follows: given a query, transform it using KLT and use the r most significant dimensions, and find which cluster the point falls into. Then read from disk the first r -KLT-dimensions of all the points that fall into that cluster. Sort these points in r -dimensional domain based on their distance from the query point. The first k points after the sorting would be the first level approximation of the answer set. There are two directions for progressive refinement of the query here. In the first direction, more dimensions of the points in the cluster could be read from disk. This would certainly improve the distance calculations, and hence the accuracy. The second orthogonal direction is to read neighboring points

that fall in other clusters. The best clusters to read are heuristically those whose centroids are closest to the query point in the r -dimensional design domain. The closest one is already read in the base layer. Reading the next best cluster also improves the accuracy, since now there are more candidate points to choose from.

One can follow any path in this progressive refinement framework. Fig. 16 illustrates an example of steps of the algorithm for a high dimensional data set, where r is taken to be 8. The 5×4 progressive refinement structure can be summarized as follows: we read 8, 16, 24, or 32 dimensions in the first refinement direction, and we read 1, 2, 3, 4, or 5 best clusters in the second refinement direction. An important detail is, however, that we have to read new dimensions of every point read so far, and when reading new clusters, you have to read up to the dimensionality reached so far. A node in the graph represents the stage where the corresponding clusters and dimensions are considered for approximate result. Since the design of the pages is not changed during the search, reaching a node by any path leads to the same portion of the data to be considered. Therefore, any path to a certain node results in the same number of page accesses and the same approximate result set. Any path in this framework can be used as an approximate k -NN algorithm. However, the chosen path may be important because the intermediate steps of various paths through the same node have different average time-accuracy tradeoffs. It is possible to choose the best path by a pre-analysis of the performance of each possible path as we will discuss in the performance evaluation section. Once a path is chosen at design time, the same path can be used for all k -NN queries. The bottom-left node in Fig. 16 represents a possible initial step of the algorithm. The top-right node illustrates the case where 32 dimensions of 5 best clusters are taken into consideration for computing the answer to the k -NN query.

Clustering based organization can also be extended for exact NN searching. By pre-computing additional information about each cluster one can compute minimum and maximum possible distances from a query to a cluster, which can then

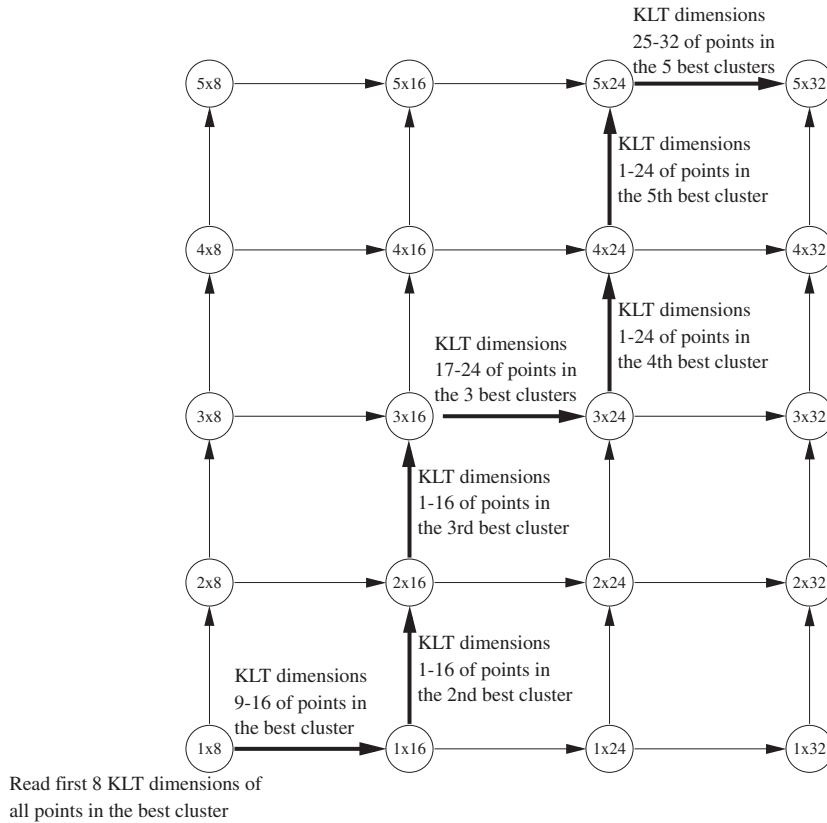


Fig. 16. A path chosen in the progressive refinement domain.

be used in an exact NN searching. A possible way is to represent each cluster by a circle with the mean as the center and the line from the mean to its furthest point as the radius, and define mindist and maxdist with respect to this circle. During the search process, clusters that have mindist greater than the maxdist of one of the clusters can be eliminated without retrieval. Similarly, once a cluster is retrieved and actual distances are computed, all clusters with maxdist greater than the current NN distance can be eliminated from the search process.

8. Performance evaluation

In this section, we evaluate the performance of the proposed techniques with the state-of-art

approaches for exact and approximate NN query processing. We first compare the performances of VA⁺-file and VA-file for exact NN queries. We then compare and analyze all discussed techniques for approximate NN searching.

8.1. Performance evaluation setup

For performance evaluation purposes, we used various real-life data sets from different application domains. The techniques proposed in this paper are for high dimensional data sets, therefore we chose our data sets to have high dimensions. The first data set is a image database with 12,000 images. Images are collected from a commercial CD-ROM and 64-dimensional *Color Histograms* are computed as feature vectors. The second data set, *Stock Time-series*, is a time-series data which

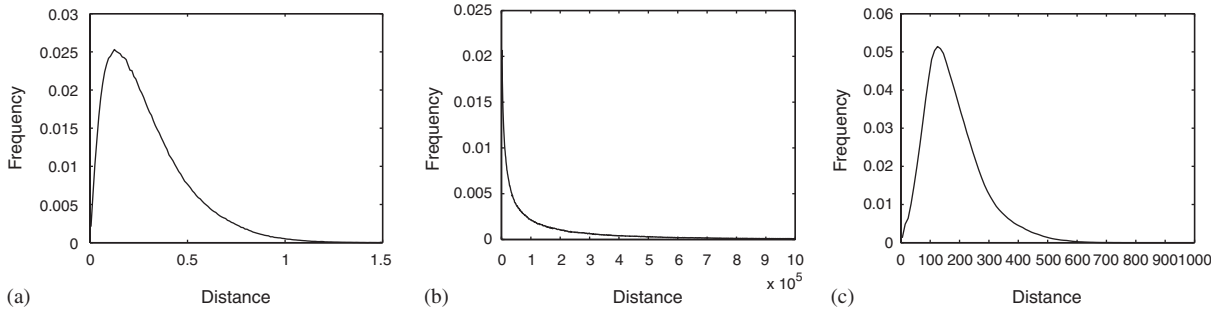


Fig. 17. Distance distribution of the data sets: (a) Color Histogram; (b) Stock Market Time-Series; (c) Satellite Image Texture.

contains 360 days stock price movements of 2000 companies, i.e., 2000 data points with dimensionality 360. The third data set is *Landsat satellite image* database with 100,000 images. Sixty-dimensional vectors representing texture features of Landsat images [25] are produced for similarity searching. Texture information of blocks of large aerial photographs are computed using Gabor filtering. This data set provides challenging problems in high dimensional indexing and is widely used in high dimensional indexing and similarity searching research [26–28]. In order to characterize each of these data sets, the frequencies of the distances between each pair of vectors for each of them are given in Fig. 17.

For the Landsat image data set, the page size is taken as 8 KB. For color histogram and time-series data sets the page capacity is assumed to be 240 floating-point numbers, i.e., approximate page size = 1 KB. The page capacities are chosen differently, since the Landsat data is much larger than the other two. We illustrate the results for 10-NN queries for all data points in the data set, i.e., query points = data points. For approximate NN searching part of the experiments, the two types of metrics discussed before are used for performance evaluation:

1. Average number of pages accessed versus average number of false hits (= false dismissals in this case).
2. Average number of pages accessed versus average ratio of total distances. Denoting by \mathbf{q} the query point, and by \mathbf{x}_j and $\tilde{\mathbf{x}}_j$ the true and

the approximate j th NN, respectively, the ratio of total distances is given by $D = \frac{\sum_{j=1}^{10} \|\mathbf{q} - \tilde{\mathbf{x}}_j\|^2}{\sum_{j=1}^{10} \|\mathbf{q} - \mathbf{x}_j\|^2}$. Now, it is easy to see that $D \geq 1$, with equality if and only if $\|\mathbf{q} - \tilde{\mathbf{x}}_j\| = \|\mathbf{q} - \mathbf{x}_j\|$ for $j = 1, \dots, 10$, which means a set of exact 10-NN is retrieved.

We note that with sequential search (of course with zero false hits and $D = 1$), one has to read around 534 pages, each 1 K size, for the image color histogram data set and 3000 pages, each 1 K size, for time-series data set. For Landsat image data set it is 2930 pages, each 8 K size.

8.2. Performance comparison of VA⁺-file and VA-file for exact k -NN queries

The total number of bits allocated to the feature vectors are kept same for both VA-file and VA⁺-file. This number can be typically decided based on the available memory. For our experiments, we have designed each of the techniques using 3, 4, 5, and 6 bits/dimension. We performed queries asking 10-NN of the data points in the data set. The same NN algorithm that was discussed before was applied to both techniques. The performance of the methods will be evaluated in terms of their first phase elimination power, and the number of vectors they have to visit in the second phase. For all cases the stated results are mean values. Better elimination power in the first phase leads to less number of candidates remaining for the second phase of the algorithm and less number

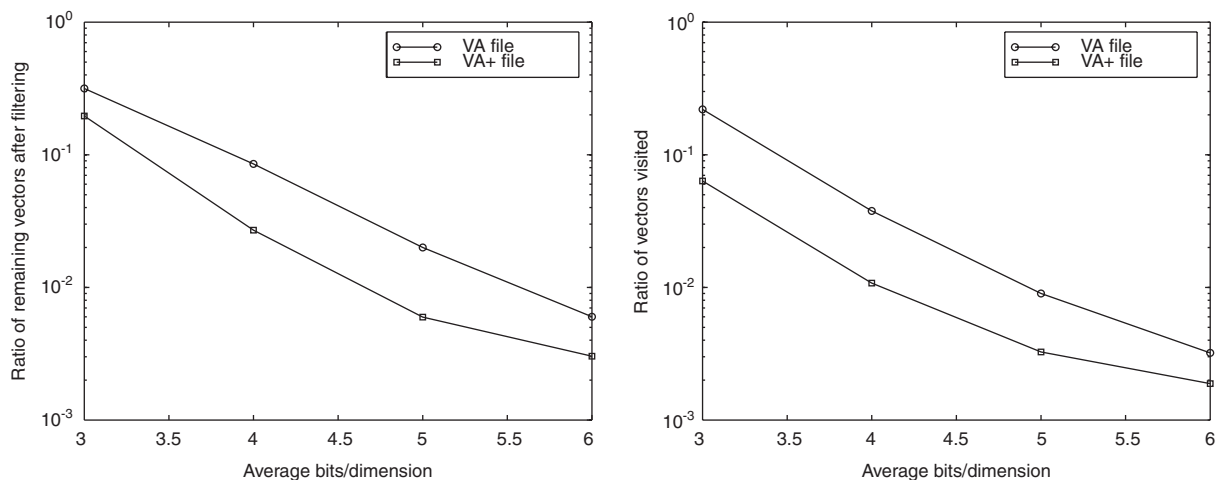


Fig. 18. Comparison between VA-file and VA⁺-file, in terms of the first (left) and second (right) phase performances, for the satellite image data.

of vectors visited in the second phase means less number of *I/O* operations needed.

Fig. 18 compares the first and second phase performances of the methods, for the satellite image data. Note that the figures are plotted on a logarithmic *y*-axis. In the first phase, for the VA-file method, there are 1.5 to 3 times as many remaining vectors after filtering as there are for the VA⁺-file method. In the second phase, the VA-file method visits 1.7–3.5 times as many vectors as the VA⁺-file method does. The results in the second phase can be seen as directly proportional to the number of page accesses and *I/O* cost during the search. Note that any vector approximation based method has to visit at least 10 vectors, i.e., 10^{-4} of all the data. For example, by just reading 9 more vectors for the 6-bit case, the VA⁺-file achieves the task, whereas the VA-file needed to visit 22 more.

Fig. 19 shows the comparison of the two methods for the image color histogram data. In the first phase, for the VA-file method, there are 1.5–5 times as many remaining vectors after filtering as there are for the VA⁺-file method. In the second phase, the VA-file method visits 2.2–8.2 times as many vectors as the VA⁺-file method does. This means that the *I/O* cost occurred during the *k*-NN searching in the VA⁺-file is much less than the previous approach. In this case, any

vector approximation based method has to visit at least 10 vectors, i.e., 8.3×10^{-4} of all the data. So by just reading 4 more vectors for the 6-bit case, the VA⁺-file achieves the task, whereas the VA-file needed to visit 20 more.

In both data sets, the KLT transforms the data set into a more suitable domain and the query speedups are achieved by the careful organization of the VA⁺-file using non-uniform bit allocation and the scalar quantization.

8.3. Performance evaluation of simple progressive techniques

We now evaluate the performance of the four simple progressive approaches discussed earlier: partial sequential scan, KLT, VA-file, and VA⁺-file based approximate *k*-NN searching. The approximate search on VA-file and VA⁺-file is implemented by omitting the second step of the exact NN search algorithm. Lower and upper bounds are used to estimate the similarity distances, and these distances are used to answer the query using only the first step. This approach is favorable since it does not need additional *I/O* operations which are required in second step of exact NN searching. Using the two metrics discussed before, we also demonstrate the feasi-

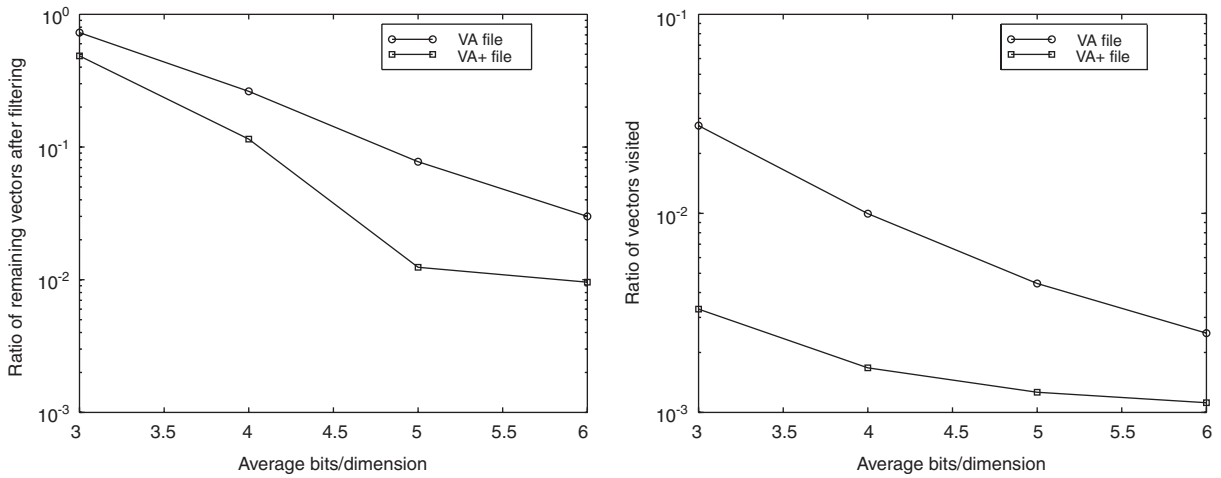


Fig. 19. Comparison between VA-file and VA⁺-file methods, in terms of the first (left) and second (right) phase performances, for the image color histogram data.

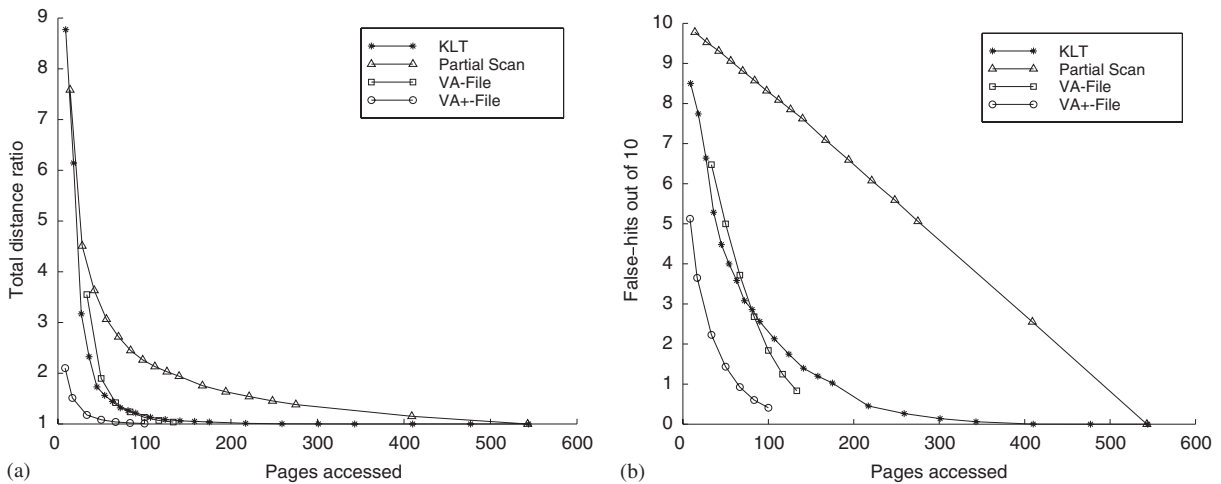


Fig. 20. Time-accuracy trade-off achieved by approximate search techniques: (a) Metric: D; (b) Metric: Number of False Hits.

bility and efficiency of *approximate k*-NN searching. The results presented in this section also establish that even simple techniques, such as KLT, can be extended and adapted for approximate searching in a very effective way. We performed experiments on various data sets and here illustrate the representative performance results for the color histogram data set, consisting of 64-dimensional image color histogram of 2000 data points. The performance behavior of the techniques is similar for all data sets.

Fig. 20 illustrates a clear demonstration of the time accuracy trade-off achieved by all approximate NN approaches. As the retrieved number of pages increases, so does the quality of the approximations. The performance of the original VA-file approach is very close to that of the simple KLT-based technique. The VA⁺-file based approximation outperforms all the methods described so far. As an example, to reach $D = 1.1$, the partial sequential scan technique retrieves approximately 4 times as many pages as the

KLT-based technique retrieves, and the KLT-based technique retrieves 2.5 times as many pages as the VA⁺-file technique does. Similarly, by reading 80 pages, the KLT-based and the partial sequential scan techniques achieve $D = 1.27$ and $D = 2.5$, respectively, whereas the VA⁺-file technique achieves $D = 1.03$. Fig. 20(b) depicts the number of false hits in each of the approximate techniques.

8.4. Performance evaluation of cluster-based approach

We implemented the proposed cluster-based technique and analyzed the performance of k -NN queries. For clarity, we illustrate the comparisons with the VA⁺-file based approximation technique since it outperformed all other methods. We summarize the results on three data sets, i.e., color histogram, stock time-series and landsat image texture. First, we illustrate the detailed performance evaluation on the image color histogram data set, then we summarize the results for the stock time-series and Landsat image texture feature data sets.

We start with the experiments on *color histogram* data. As described before, first, a dimensionality reduction (from 64 to 8) is performed on each point in the data set using KLT. We implemented the modified K -means clustering algorithm to create the clusters and the corresponding cluster centroids. Four different minimum cluster sizes are selected: 10, 20, 30, and 40. The number of clusters corresponding to these four scenarios are: 94, 47, 34, and 24, respectively. Since the representative dimensionality is 8, the amount of storage these representatives need is only 0.5%, 0.29%, 0.2%, and 0.15% of the data set. Decreasing the representative dimensionality or increasing the minimum cluster size results in less storage needs for the representatives. This percentage is much less for larger data sets or data sets with higher dimensionalities, such as the Landsat image feature and the time-series data sets that we use later in our performance evaluation. The largest ratio for the additional storage needed for representatives is only 0.029% of the entire time-series data set and 0.026% of the Landsat image

Table 1
Time-accuracy performance achieved by the clustering with minimum cluster size 10

# Clusters × dimensions	# Pages accessed	# False hits	D
1 × 8	1.0000	4.8410	1.5669
2 × 8	2.0000	3.7195	1.3935
3 × 8	3.0000	3.3420	1.3502
4 × 8	4.0000	3.1965	1.3333
5 × 8	5.0000	3.1365	1.3229
1 × 16	2.0000	4.5020	1.3730
2 × 16	4.0000	2.8320	1.1638
3 × 16	6.0000	2.1190	1.1104
4 × 16	8.0000	1.7890	1.0885
5 × 16	10.0000	1.6335	1.0777
1 × 24	3.0000	4.4530	1.3415
2 × 24	6.0000	2.6035	1.1287
3 × 24	9.0000	1.7110	1.0728
4 × 24	12.0000	1.2375	1.0490
5 × 24	15.0000	0.9825	1.0380
1 × 32	4.0000	4.4450	1.3258
2 × 32	8.0000	2.5530	1.1112
3 × 32	12.0000	1.6015	1.0536
4 × 32	16.0000	1.0795	1.0305
5 × 32	20.0000	0.7795	1.0200

data set. Since the size of the representative set is very small, the necessary processing on the representatives is also negligible compared to the entire data set.

We analyzed the performance by using the 5×4 progressive refinement discussed before. We read 8, 16, 24, or 32 dimensions in the first refinement direction, and we read 1, 2, 3, 4, or 5 best clusters in the second refinement direction.

The performance of the algorithm with minimum cluster size 10 is presented in Table 1. The path indicated by the rows with bold characters in Table 1, i.e., the path 1 × 8, 1 × 16, 2 × 16, 3 × 16, 3 × 24, 4 × 24, 4 × 32, and 5 × 32, corresponds to the path illustrated in Fig. 16, and is actually the best path according to the error measure D . This path can be found by using dynamic programming technique on possible paths in the progressive refinement framework. The goal is to achieve best accuracy by retrieving minimal number of pages. The same performance results, in comparison with those of the VA⁺-file based technique, are plotted

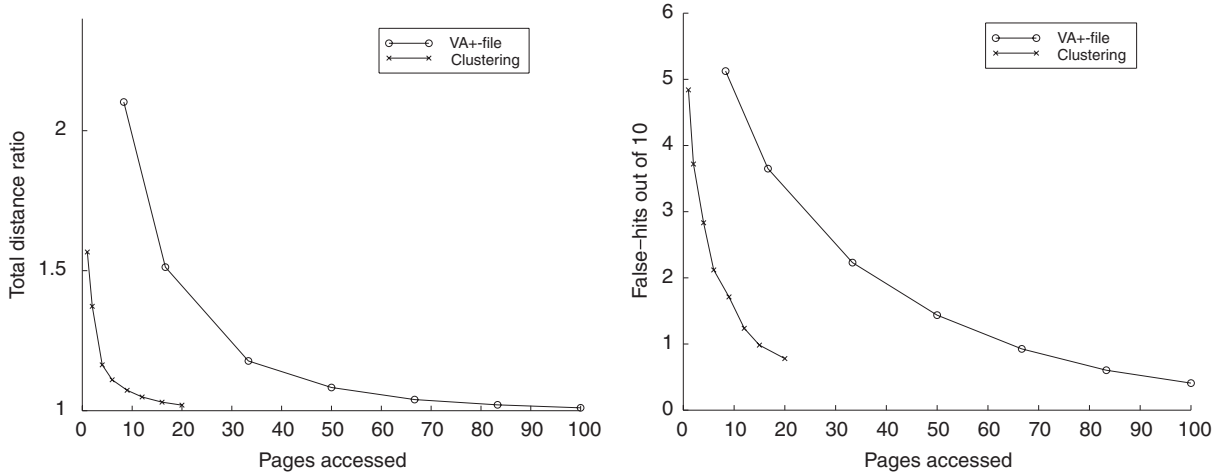


Fig. 21. Comparison of time-accuracy performances achieved by the cluster-based technique (with minimum cluster size 10) and VA⁺-file.

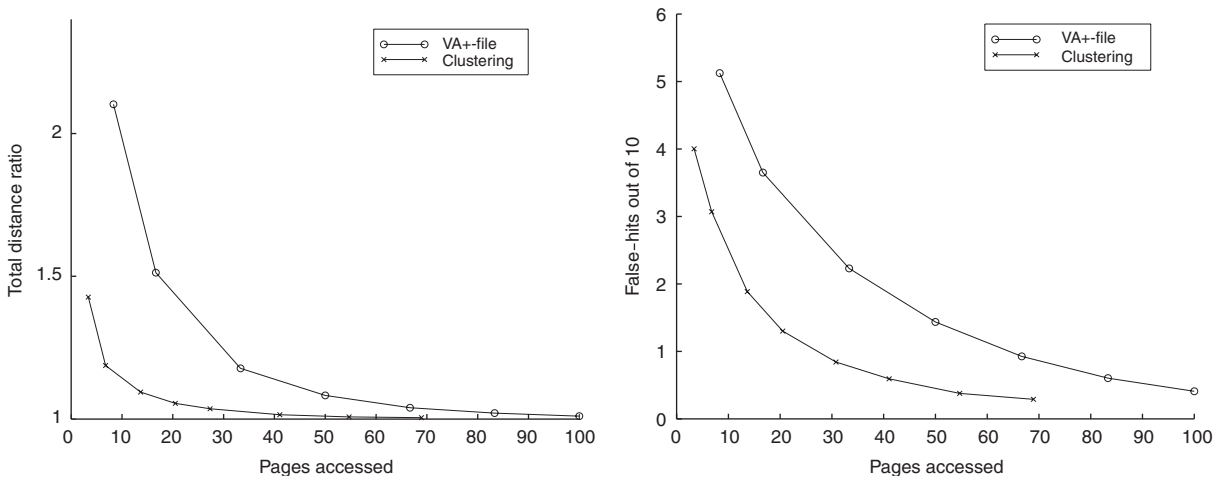


Fig. 22. Comparison of time-accuracy performances achieved by the cluster-based technique (with minimum cluster size 40) and VA⁺-file.

in Fig. 21. In the figures, only the best paths are considered for clarity.

As an illustrative example, by reading 16 pages, the cluster-based algorithm achieves an error of $D = 1.03$, whereas the VA⁺-file technique reaches an error level of $D = 1.56$. Similarly, to achieve the accuracy that the clustering based technique achieves reading 16 pages, the VA⁺-file technique reads about 75 pages. The cluster-based technique achieves speedup from 4 to 15 compared to the VA⁺-file technique.

As mentioned before, the size of the representatives is negligible compared to the size of the data set. The overhead due to the representatives is very small: 4, 2, 2, and 1 additional page accesses for cluster sizes 10, 20, 30 and 40, respectively.

The performance results are very similar for various minimum cluster sizes. Fig. 22 illustrates the comparison of time-accuracy performances achieved by the VA⁺-file based technique and clustering with minimum cluster size 40. The figure is again drawn for the best path. The detailed

Table 2
Time-accuracy performance achieved by the clustering with minimum cluster size 40

# Clusters \times dimensions	# Pages accessed	# False hits	D
1 \times 8	3.3855	4.0025	1.4272
2 \times 8	6.8345	3.3000	1.3409
3 \times 8	10.2640	3.1525	1.3252
4 \times 8	13.6695	3.1065	1.3207
5 \times 8	17.2325	3.0910	1.3202
1 \times 16	6.7710	3.0690	1.1877
2 \times 16	13.6690	1.8865	1.0943
3 \times 16	20.5280	1.5670	1.0737
4 \times 16	27.3390	1.4555	1.0664
5 \times 16	34.4650	1.4240	1.0644
1 \times 24	10.1565	2.7780	1.1518
2 \times 24	20.5035	1.3015	1.0550
3 \times 24	30.7920	0.8430	1.0339
4 \times 24	41.0085	0.6695	1.0259
5 \times 24	51.6975	0.5980	1.0234
1 \times 32	13.5420	2.6980	1.1339
2 \times 32	27.3380	1.1125	1.0360
3 \times 32	41.0560	0.5935	1.0151
4 \times 32	54.6780	0.3785	1.0075
5 \times 32	68.9300	0.2900	1.0049

performance of the algorithm with minimum cluster size 40 is presented in Table 2.

Fig. 23 illustrates the comparison of time-accuracy performances achieved by the clustering-based approach with minimum cluster sizes 10, 20, 30, and 40.

We also analyze the scalability of our techniques as the data set size grows. We evaluate the performance of full and partial sequential search, KLT-based dimensionality reduction, VA⁺-file, and clustering based technique by comparing the number of pages accessed as the data set size grows. Fig. 24 illustrates the comparison of five techniques. The curves for sequential scan, KLT, VA⁺-file, and clustering based techniques represent the result set with a relative distance of $D = 1.05$, where $D = 1$ represents the actual result without error. The VA⁺-file based approach achieves significant speedup compared to the full and partial sequential scan, and the KLT-based technique. The clustering based approach achieves significant speedups compared to the other two

techniques and scales much better as the data size grows. For the data set with 1000 data objects, our cluster-based technique accesses 30 times less pages than sequential scan, 29 times less than the partial sequential scan, 9.3 times less than KLT-based technique, and finally 3.5 times less than the VA⁺-file approach access for $D = 1.05$. As the data size increases, the speedups achieved by our techniques increase. For example, for 3000 data objects and $D = 1.05$, our cluster-based technique achieves speedups 51.1, 47.3, 16.5, and 6.2 compared to full sequential scan, partial sequential scan, KLT-based, and VA⁺-file based techniques, respectively.

The speedups are higher for the case $D = 1.1$, i.e., 49.8, 44, 11.1, and 4.32 times better than full sequential scan, partial sequential scan, KLT-based, and VA⁺-file techniques, respectively. Again as the data size increases, the speedups achieved by our techniques increase. For 3000 data objects and $D = 1.1$, the cluster-based technique achieves speedups 89.4, 75.2, 22.2, and 8.1 compared to full sequential scan, partial scan, KLT-based, and VA⁺-file based techniques, respectively. We repeated the similar experiments and got similar results for the metric that considers false hits. Table 3 illustrates the results of all these experiments, i.e., the number of page accesses in 10-NN queries on data sets containing 1000, 2000, and 3000 data points, for D to be 1.05 and 1.1, and for number of false hits to be 1 and 1.5.

For the 360-dimensional time-series data set, the performance comparison of the techniques is very similar to the performance comparison for the image color histogram data set. We note that, for the time-series data set, the speedups achieved by our technique are more than the speedups that we obtained for image feature data set. Fig. 25 illustrates a sample performance comparison of the VA⁺-file and our clustering based technique for both metrics, i.e., D and the number of false hits. Some representative numbers from these experiments are as follows. To achieve $D = 1.1$, our clustering based technique reads an average of 4 pages whereas VA-file reads an average of more than 25 pages. Similarly, for $D = 1.05$, the clustering based technique reads an average of 5.1 pages, whereas VA-file reads average of 43.7

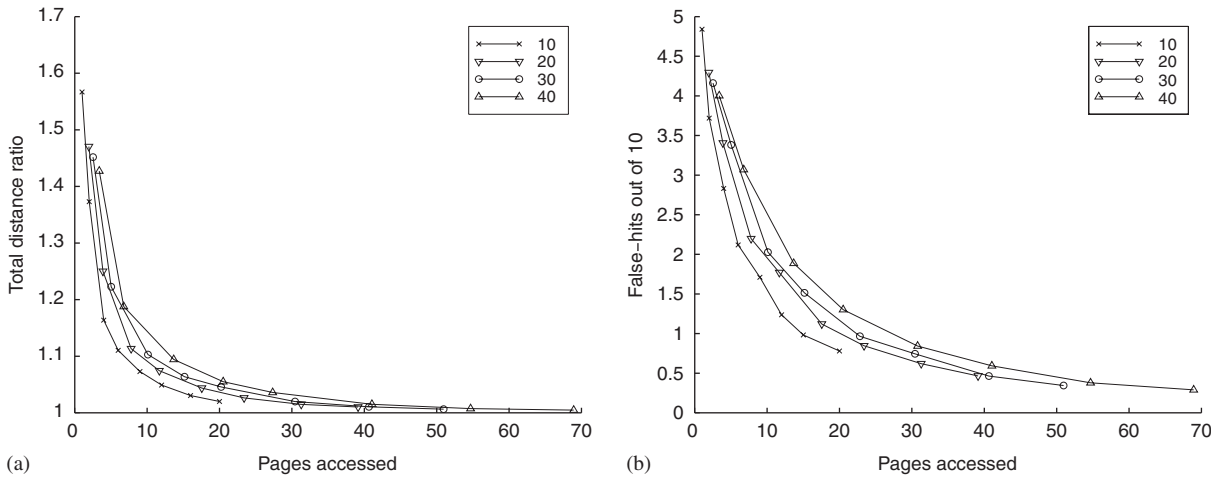


Fig. 23. Comparison of time-accuracy performances achieved by the clustering with minimum cluster sizes 10, 20, 30, and 40: (a) Metric: D ; (b) Metric: Number of False Hits.

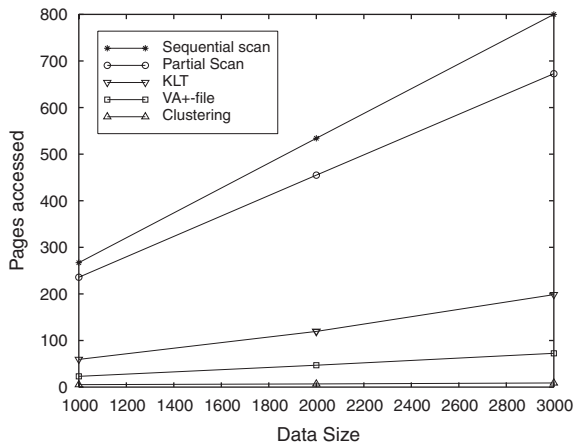


Fig. 24. Comparison of number of pages accessed among sequential scan, partial scan, KLT, VA^+ -file, and cluster-based approximations, as the data set size grows linearly. All the curves are for $D = 1.05$.

pages in each query. The speedup achieved for $D = 1.1$ is 6.3, and the speedup achieved for $D = 1.05$ is 8.5. These speedups were less for the image color histogram data set, i.e., 5.3 and 6.8, respectively.

We repeated the experiments for the Landsat image data set which contains 100,000 60-dimensional texture feature vectors. Since the data set size is much larger, the page size is taken as 8 KB.

Table 3

Comparison of number of pages accessed for data sets with 1000, 2000, and 3000 data objects

Data size (# of pages)	1000	2000	3000
EXACT	267.00	534.00	800.00
SEQ. (partial) $D = 1.05$	255.19	498.99	740.89
SEQ. (partial) $D = 1.1$	235.75	454.99	672.78
KLT $D = 1.05$	81.31	160.32	258.72
KLT $D = 1.1$	59.41	119.50	198.39
VA^+ -FILE $D = 1.05$	30.67	62.73	96.94
VA^+ -FILE $D = 1.1$	23.19	47.01	72.61
CLUSTER $D = 1.05$	8.78	11.87	15.65
CLUSTER $D = 1.1$	5.36	6.83	8.95
EXACT	267.00	534.00	800.00
SEQ. (partial) $F = 1$	250.12	490.47	729.67
SEQ. (partial) $F = 1.5$	237.01	464.21	690.00
KLT $F = 1$	82.28	177.02	274.24
KLT $F = 1.5$	62.68	135.97	222.66
VA^+ -FILE $F = 1$	30.92	64.23	100.85
VA^+ -FILE $F = 1.5$	23.23	48.65	77.92
CLUSTER $F = 1$	11.37	14.79	24.06
CLUSTER $F = 1.5$	7.26	10.10	13.75

Metrics: left table: D , right table: false hits out of 10.

The percentage of storage needed for the representative set is only 0.026% of the data set. In general, the speedups that are achieved for this data set are better than the color histogram data set and comparable with the time-series data set.

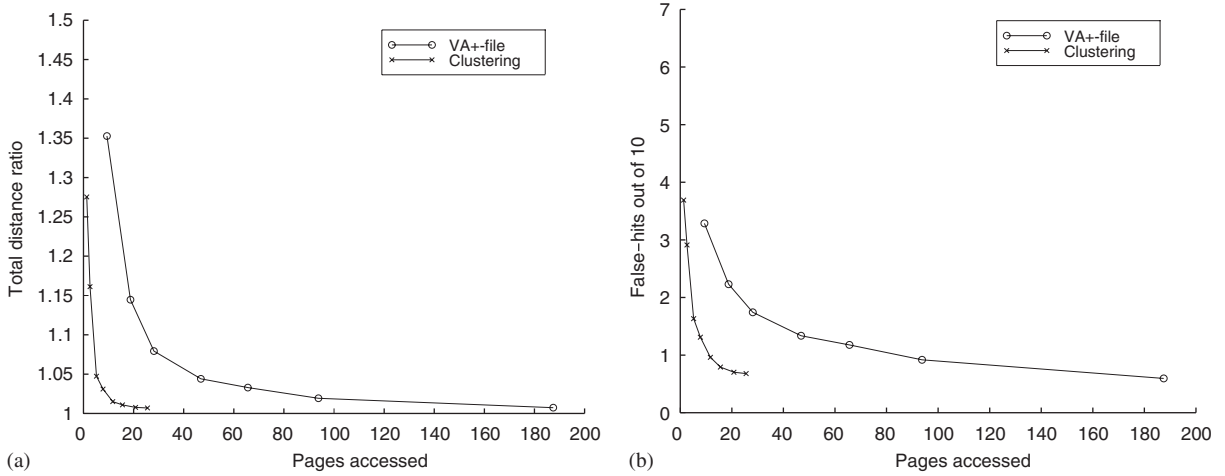


Fig. 25. Comparison of time-accuracy performances achieved by the clustering based technique and VA⁺-file on time-series data set: (a) Metric: D; (b) Metric: Number of False Hits.

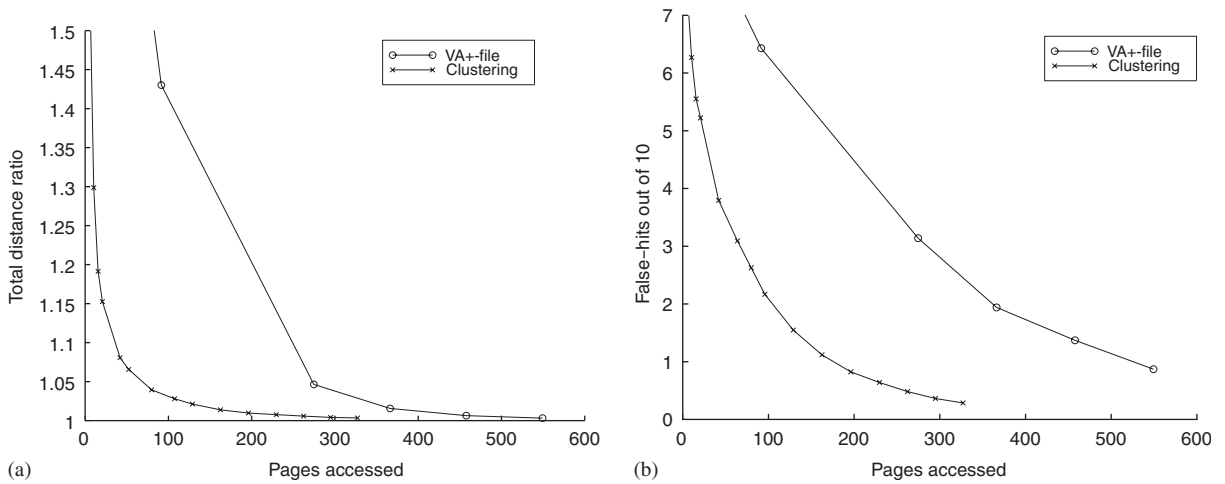


Fig. 26. Comparison of time-accuracy performances achieved by the clustering based technique and VA⁺-file on Landsat image data set: (a) Metric: D; (b) Metric: Number of False Hits.

The results show that our clustering based technique maintains its superiority also for large data sets. Fig. 26 illustrates a sample performance comparison of the VA⁺-file and our clustering based technique for both metrics, i.e., D and the number of false hits. Some representative numbers from these experiments are as follows. To achieve $D = 1.1$, our clustering based technique reads an average of 36 pages where VA-file reads an average 249 pages. Similarly, for $D = 1.05$, the

clustering based technique reads an average of 68 pages, where VA-file reads average of 273 pages in each query. The speedup achieved for $D = 1.05$ is 4, and the speedup achieved for $D = 1.1$ is 6.9.

9. Conclusion

Similarity searching in high dimensional spaces has several applications in modern databases.

Extensions of traditional techniques for low dimensional data lose its effectiveness after a small number of dimensions. Geometric imagination of high dimensional data is impossible, therefore simple intuition does not work. Therefore, we focused on building techniques that are specifically built for high dimensional data sets. The techniques proposed in this paper are built to work for real-life data sets, which have non-uniform distribution. VA-file [16] was one of the first attempts for developing a technique specific to high dimensional searching. However, VA-file is more effective when the data is uniformly distributed. Real data sets are not uniformly distributed, are often clustered, and the dimensions of the feature vectors in real data sets are usually correlated. Based on analysis of high dimensional data sets, we proposed VA⁺-file, a scalar quantization based technique for similarity searching on high dimensional data sets [60].

We also discussed how to improve searching and make it progressive by allowing some approximations in the query result. We developed a general framework for approximate NN queries. The current approaches for NN query processing can be categorized based on either their ability to reduce the data set that needs to be examined, or their ability to reduce the information of each data object. We first proposed modifications to well-known techniques to support progressive approximate searching, so that a user may stop the retrieval process once enough information has been returned. We then developed a new technique based on clustering that merges the benefits of the two general classes of approaches. Our cluster-based approach allows a user to progressively explore the approximate results with increasing accuracy [61]. We proposed a new metric for evaluation of approximate NN searching techniques. Using both the proposed and the traditional metrics, an extensive experimental evaluation is performed on several real-life data sets. Our performance evaluation establishes that the proposed techniques achieves very significant speedups over the existing techniques for high dimensional similarity searching.

Acknowledgements

This material was prepared with the support of the US Department of Energy (DOE) Award No. DE-FG02-03ER25573. However, any opinions, findings, conclusions or recommendations expressed herein are those of the author, and do not necessarily reflect the views of the DOE.

References

- [1] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, W. Equitz, Efficient and effective querying by image content, *J. Intell. Inf. Syst.* 3 (1994) 231–262.
- [2] S. Berchtold, H.-P. Kriegel, S3: similarity search in CAD database systems, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, 1997, pp. 564–567.
- [3] X. Cheng, R. Dolin, M. Neary, S. Prabhakar, K. Ravikanth, D. Wu, D. Agrawal, A. El Abbadi, M. Freeston, A. Singh, T. Smith, J. Su, Scalable access within the context of digital libraries, in: *IEEE Proceedings of the International Conference on Advances in Digital Libraries*, ADL, Washington, DC, 1997, pp. 70–81.
- [4] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, Fast subsequence matching in time-series databases, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Minneapolis, May 1994, pp. 419–429.
- [5] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, Z. Protopapas, Fast nearest neighbor search in medical image databases, in: *Proceedings of the International Conference on Very Large Data Bases*, Mumbai, India, 1996, pp. 215–226.
- [6] H. Samet, *The Design and Analysis of Spatial Structures*, Addison-Wesley Publishing Company, Inc., Massachusetts, 1989.
- [7] V. Gaede, O. Gunther, Multidimensional access methods, *ACM Comput. Surv.* 30 (1998) 170–231.
- [8] J. Nievergelt, H. Hinterberger, K.C. Sevcik, The grid file: an adaptable, symmetric multikey file structure, *ACM Trans. Database Syst.* 9 (1) (1984) 38–71.
- [9] J.T. Robinson, The kdb-tree: a search structure for large multi-dimensional dynamic indexes, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1981, pp. 10–18.
- [10] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1984, pp. 47–57.
- [11] N. Beckmann, H. Kriegel, R. Schneider, B. Seeger, The R* tree: an efficient and robust access method for points and rectangles, in: *Proceedings of the ACM SIGMOD*

- International Conference on Management of Data, pp. 322–331, May 23–25, 1990.
- [12] D.B. Lomet, B. Salzberg, The hb-tree: a multi-attribute indexing method with good guaranteed performance, *ACM Trans. Database Syst.* 15 (4) (1990) 625–658.
- [13] D. White, R. Jain, Similarity indexing with the SS-tree, in: *Proceedings of the International Conference on Data Engineering*, 1996, pp. 516–523.
- [14] K. Chakrabarti, S. Mehrotra, The hybrid tree: an index structure for high dimensional feature spaces, in: *Proceedings of the International Conference Data Engineering*, Sydney, Australia, 1999, pp. 440–447.
- [15] S. Berchtold, C. Böhm, D. Keim, H. Kriegel, A cost model for nearest neighbor search in high-dimensional data space, in: *Proceedings of the ACM Symposium on Principles of Database Systems*, Tuscon, Arizona, June 1997, pp. 78–86.
- [16] R. Weber, H.-J. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, in: *Proceedings of the International Conference on Very Large Data Bases*, New York City, New York, August 1998, pp. 194–205.
- [17] S. Berchtold, C. Böhm, H. Jagadish, H. Kriegel, J. Sander, Independent quantization: an index compression technique for high-dimensional data spaces, in: *Proceedings of the 16th International Conference on Data Engineering*, San Diego, CA, 2000, pp. 577–588.
- [18] Y. Sakurai, M. Yoshikawa, S. Uemura, H. Kojima, The a-tree: an index structure for high-dimensional spaces using relative approximation, in: *Proceedings of 26th International Conference on Very Large Data Bases*, September 10–14, Cairo, Egypt, 2000, pp. 516–526.
- [19] A. Gersho, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, MA, 1992.
- [20] H. Karhunen, Über lineare methoden in der wahrscheinlichkeitsrechnung, *Ann. Acad. Sci. Fenn.* 37 (1) (1947) 3–79.
- [21] M. Loeve, *Fonctions aleatoires de seconde ordre, Procesus Stochastiques et Mouvement Brownien*, 1948.
- [22] S.P. Lloyd, Least squares quantization in pcm, *IEEE Trans. Inf. Theory* 28 (1982) 127–135.
- [23] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design, *IEEE Trans. Commun.* 28 (1980) 84–95.
- [24] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: *Proceedings of the Fifth Berkeley Symposium on Mathematics Statistics and Probability*, vol. 1, 1967, pp. 281–297.
- [25] B.S. Manjunath, W.Y. Ma, Texture features for browsing and retrieval of image data, *IEEE Trans. Pattern Anal. Mach. Intell.* 18 (8) (1996) 837–842.
- [26] B.S. Manjunath, Airphoto dataset, May 2000, <http://vivaldi.ece.ucsb.edu/Manjunath/research.htm>.
- [27] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: *Proceedings of the International Conference on Very Large Data Bases*, Edinburgh, Scotland, UK, September 1999, pp. 518–529.
- [28] P. Ciaccia, M. Patella, PAC nearest neighbor queries: approximate and controlled search in high-dimensional and metric spaces, in: *Proceedings of the International Conference Data Engineering*, San Diego, California, March 2000, pp. 244–255.
- [29] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, The QBIC project: querying images by content using color, texture and shape, in: *Proceedings of the SPIE Conference 1908 on Storage and Retrieval for Image and Video Databases*, vol. 1908, 1993, pp. 173–187.
- [30] V.S. Subrahmanian, *Principles of Multimedia Database Systems*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1999.
- [31] R. Agrawal, C. Faloutsos, A. Swami, Efficient similarity search in sequence databases, in: *Fourth International Conference on Foundations of Data Organization and Algorithms*, 1993, pp. 69–84.
- [32] K. Lin, H.V. Jagadish, C. Faloutsos, The TV-tree: an index structure for high-dimensional data, *VLDB J.* 3 (1995) 517–542.
- [33] S. Berchtold, D. Keim, H. Kriegel, The X-tree: an index structure for high-dimensional data, in: *Proceedings of the International Conference on Very Large Data Bases*, Bombay, India, 1996, pp. 28–39.
- [34] S. Berchtold, C. Böhm, H.-P. Kriegel, The Pyramid-Technique: towards breaking the curse of dimensionality, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, USA, June 1998, pp. 142–153.
- [35] N. Roussopoulos, S. Kelly, F. Vincent, Nearest neighbor queries, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 1995, pp. 71–79.
- [36] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, An optimal algorithm for approximate nearest neighbor searching, in: *Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994, pp. 573–582.
- [37] S. Arya, D.M. Mount, Approximate range searching, in: *Eleventh Annual Symposium on Computational Geometry*, June 1995, pp. 172–181.
- [38] R. Weber, K. Böhm, Trading quality for time with nearest-neighbor search, in: *Proceedings of the International Conference on Extending Database Technology*, Konstanz, Germany, March 2000, pp. 21–35.
- [39] P. Zezula, P. Savino, G. Amato, F. Rabitti, Approximate similarity retrieval with M-trees, *The VLDB J.* 4 (1998) 275–293.
- [40] O. Egecioglu, H. Ferhatosmanoglu, Dimensionality reduction and similarity distance computation by inner product approximations, in: *Proceedings of the Ninth ACM International Conference on Information and Knowledge Management*, McLean, Virginia, November 2000, pp. 219–226.
- [41] U. Ogras, H. Ferhatosmanoglu, Dimensionality reduction using magnitude and shape approximations, in: *Proceedings of the Conference on Information and Knowledge*

- Management, New Orleans, LA, November 2003, pp. 91–98.
- [42] J. Goldstein, R. Ramakrishnan, Contrast plots and P-sphere trees: space vs. time in nearest neighbor searches, in: Proceedings of the VLDB '00 Conference, Cairo, Egypt, 2000, pp. 429–440.
- [43] T. Seidl, H.-P. Kriegel, Efficient user-adaptable similarity search in large multimedia databases, in: Proceedings of the International Conference on Very Large Data Bases, Athens, Greece, 1997, pp. 506–515.
- [44] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is “nearest neighbor” meaningful, in: International Conference on Database Theory, Jerusalem, Israel, January 1999, pp. 217–225.
- [45] A.V. Oppenheim, R.W. Schaffer, Discrete-Time Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- [46] T. Kailath, Modern Signal Processing, Springer, Berlin, 1985.
- [47] K.R. Castleman, Digital Image Processing, Prentice-Hall, Englewood Cliffs, NJ, 1996.
- [48] C. Yu, B.C. Ooi, K. Tan, H.V. Jagadish, Indexing the distance: an efficient method to knn processing, in: VLDB, 2001, pp. 421–430.
- [49] N. Koudas, B.C. Ooi, H.T. Shen, A.K.H. Tung, Ldc: enabling search by partial distance in a hyper-dimensional space, in: ICDE, 2004, pp. 6–17.
- [50] E.J. Keogh, K. Chakrabarti, S. Mehrotra, M.J. Pazzani, Locally adaptive dimensionality reduction for indexing large time series databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, Santa Barbara, CA, 2001, pp. 151–162.
- [51] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, N. Koudas, Non-linear dimensionality reduction techniques for classification and visualization, in: The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, July 2002.
- [52] D. Hull, Improving text retrieval for the routing problem using latent semantic indexing, in: Proceedings of the 17th ACM-SIGIR Conference, 1994, pp. 282–291.
- [53] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Launder, R. Harshman, Indexing by latent semantic analysis, J. Am. Soc. Inf. Sci. 41 (1990) 391–407.
- [54] S.T. Dumais, Improving the retrieval of information from external sources, Behav. Res. Methods, Instrum. Comput. 23 (1991) 229–236.
- [55] K.V.R. Kanth, D. Agrawal, A. Singh, Dimensionality reduction for similarity searching in dynamic databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, Washington, June 1998, pp. 166–176.
- [56] A.P. de Vries, N. Mamoulis, N. Nes, M.L. Kersten, Efficient k -NN search on vertically decomposed data, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2002, pp. 322–333.
- [57] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
- [58] R.M. Haralick, L.G. Shapiro, Image segmentation techniques, Comput. Vision Graphics Image Process. 29 (1) (1985) 100–132.
- [59] J. Lin, M. Vlachos, E. Keogh, D. Gunopulos, Iterative incremental clustering of time series, in: Ninth Conference on Extending Database Technology (EDBT 2004), Crete, Greece, March 2004.
- [60] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, A. El Abbadi, Vector approximation based indexing for non-uniform high dimensional data sets, in: Proceedings of the Ninth ACM International Conference on Information and Knowledge Management, McLean, Virginia, November 2000, pp. 202–209.
- [61] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, A. El Abbadi, Approximate nearest neighbor searching in multimedia databases, in: Proceedings of the 17th IEEE International Conference on Data Engineering (ICDE), Heidelberg, Germany, April 2001, pp. 503–511.