Trading Quality for Time with Nearest-Neighbor Search

Roger Weber and Klemens Böhm

Institute of Information Systems
ETH Zentrum, 8092 Zurich, Switzerland
{weber,boehm}@inf.ethz.ch

Abstract. In many situations, users would readily accept an approximate query result if evaluation of the query becomes faster. In this article, we investigate approximate evaluation techniques based on the VA-File for Nearest-Neighbor Search (NN-Search). The VA-File contains approximations of feature points. These approximations frequently suffice to eliminate the vast majority of points in a first phase. Then, a second phase identifies the NN by computing exact distances of all remaining points. To develop approximate query-evaluation techniques, we proceed in two steps: first, we derive an analytic model for VA-File based NN-search. This is to investigate the relationship between approximation granularity, effectiveness of the filtering step and search performance. In more detail, we develop formulae for the distribution of the error of the bounds and the duration of the different phases of query evaluation. Based on these results, we develop different approximate query evaluation techniques. The first one adapts the bounds to have a more rigid filtering, the second one skips computation of the exact distances. Experiments show that these techniques have the desired effect: for instance, when allowing for a small but specific reduction of result quality, we observed a speedup of 7 in 50-NN search.

1 Introduction

In the presence of large collections of multimedia objects, notably images or videos, similarity search is an issue of outmost importance. Nearest-neighbor search (NN-search) in high-dimensional spaces is a common implementation of similarity search. The problem with NN-search is its performance, in particular in high-dimensional vector spaces. In such spaces, the complexity of NN-search is linear in the number N of objects to be searched. A number of experiments, whose original intention was to achieve a better performance with trees as index structures, show this [5, 4, 14]. [15, 6] come to the same conclusion using a more formal line of argumentation. Thus, we for our part have developed a data structure for NN-search, the so-called *VA-File* [15], which is not based on trees. Its efficiency is due to its use of approximations. A so-called *approximation file* contains an approximation of each vector. VA-File based NN-search consists of two steps, subsequently referred to as *phases*: the first phase computes an upper and a lower bound on the distance of each vector. This phase returns the vectors with the smallest bounds, the so-called *candidates*. The second phase retrieves the exact position of the candidates to identify the nearest neighbor. Obviously, the duration of both phases depends on the quality of the approximations.

The objective of this article is to achieve a significant performance gain by resorting to approximate results to NN-queries. Trading result quality for reduced query execution time is completely natural in the context of interactive similarity search: this kind of search typically consists of several steps. Frequently, the result of the initial query does not cover the information need of the user because the query is not *well-posed*. The subsequent steps work with refined queries, e.g. by exploiting feedback information given by the user. To support this interaction, it is crucial to have fast results that may be approximate, rather than long evaluation times.

As a main contribution of this article, we present new approximate evaluation techniques for NN-queries and investigate their performance gain as well as their loss of quality. We do so for one specific data structure, the VA-File. To this end, we proceed as follows:

As a first step, we develop an analytic model for VA-File based search. The motivation is to investigate the relation between the granularity of approximations, i.e. the number of bits used, the effectiveness of the filtering step, and the search performance. We do so by deriving the distribution of the error of the bounds. An important insight is that the bounds noticeably deviate from the distance in the vast majority of cases, i.e., there are hardly any bounds that are 'very close' to the distance. Another relationship derived in this step is the one between the physical structure of the approximation file, i.e., the number of bits per dimension, and the duration of the two phases.

As a second step, we describe two orthogonal approaches to trade result quality for query execution time with the VA-File. The approaches are as follows:

(1) As the bounds significantly deviate from the real distance in almost all cases, we can compute new bounds that are closer to each other. With these new bounds, there are hardly any 'false drops'. This facilitates a coarser approximation file that accelerates the first phase, but that leaves the number of candidates unchanged.
(2) We completely omit the second phase and estimate the distances from the lower and upper bounds.

In the following, Section 2 introduces our notation, and reviews the VA-File. In Section 3, we derive measures for the quality of approximation and the cost of VA-File based search. The findings in this section lead to approximate query evaluation techniques described in Section 4. Section 5 defines different measures for the loss of result quality. We use these measures in Section 6 to evaluate the different techniques. Section 7 discusses related work, and Section 8 concludes.

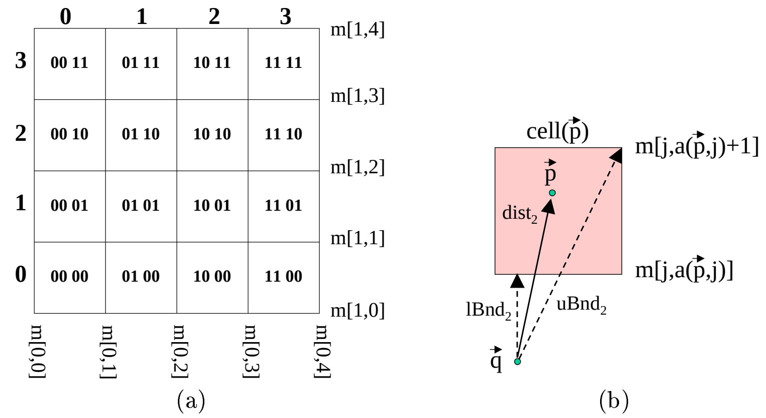


Fig. 1. Illustration of the VA-File

2 Preliminaries

To ease presentation, we explicitly consider only the L_2 distance metric (Euclidean distance). However, it is straightforward to establish the formulae in this paper for other metrics such as L_1 or L_∞ as well.

2.1 On the Relationship between Similarity Search and NN-search

Nearest-neighbor search (NN-search) in high-dimensional spaces is a common implementation of similarity search. This implementation assumes that a set of features describes each object. A vector represents the features of each object, and each feature type corresponds to a certain range of vector dimensions. E.g., Dimensions 1–16 describe the shape of the objects, Dimensions 17–32 describe the color etc. A NN-query also has such a vector representation. In our terminology, the *distance* of a vector is its distance to the query vector. NN-search (k-NN search) then means finding the (k) vector(s) with the smallest distance.

2.2 NN-Query Evaluation with the VA-File

Structure. The VA-File consists of a vector file and a file containing approximations of all vectors. The approximations are based on a grid that partitions the data space. The grid has 2^{b_j} intervals along dimension j. The intervals of this grid have numbers from 0 to $2^{b_j} - 1$ (cf. Figure 1 (a)), and the partition points $m[j, 0], \ldots, m[j, 2^{b_j}]$ bound them. These points are determined such that each interval contains the same number of vectors. Given a point $\boldsymbol{p}$, $a(\boldsymbol{p}, j)$ denotes the interval in dimension j into which $\boldsymbol{p}$ falls. A *bit-string* of length $b = \sum_{j=0}^{d-1} b_j$ represents each cell. Finally, the approximation of $\boldsymbol{p}$ is the bit-string of the cell that contains $\boldsymbol{p}$.

Bounds on the Distance between Query and Data Point. Given the approximation of a data point, we can lower and upper bound the distance between the data point $\boldsymbol{p}$ and the query point $\boldsymbol{q}$ (see Figure 1 (b)):

$$lBnd_2(\boldsymbol{p},\boldsymbol{q}) \leq dist_2(\boldsymbol{p},\boldsymbol{q}) \leq uBnd_2(\boldsymbol{p},\boldsymbol{q}) \tag{1}$$

The lower bound $lBnd_2(\boldsymbol{p},\boldsymbol{q})$ is the distance between the query point and the closest point of the cell of $\boldsymbol{p}$. Analogously, we obtain the upper bound $uBnd_2(\boldsymbol{p},\boldsymbol{q})$.

Conditions for the NN. The following two observations are important in the course of this article: a necessary condition for a point to be the NN is that its lower bound is smaller than the NN-distance.

$$\boldsymbol{p} = nn(\boldsymbol{q}) \quad \Rightarrow \quad lBnd_2(\boldsymbol{p},\boldsymbol{q}) \leq nn^{dist}(\boldsymbol{q}) \tag{2}$$

Hence, to find the NN, we only consider data points with a smaller lower bound than $nn^{dist}(\boldsymbol{q})$, the *candidates*. Moreover, if the following inequality holds for two points $\boldsymbol{p}$ and $\boldsymbol{p'}$, $\boldsymbol{p'}$ cannot be the NN.

$$uBnd_2(\boldsymbol{p},\boldsymbol{q}) < lBnd_2(\boldsymbol{p'},\boldsymbol{q}) \quad \Rightarrow \quad \boldsymbol{p'} \neq nn(\boldsymbol{q}) \tag{3}$$

NN-Search. The search algorithm has two phases: the first phase (filtering step) scans the approximations of the vectors to determine lower and upper bounds on the distance between the current vector and $\boldsymbol{q}$. Using Equation (3), we can prune vectors: if the lower bound of a vector is larger than the smallest upper bound seen so far, the vector cannot be the NN. In the second phase (refinement step), we visit the remaining vectors sorted by the lower bounds in increasing order. Due to Equation (2), this phase ends if a lower bound is encountered that is larger than the smallest distance seen so far. Hence, the algorithm visits exactly those points whose lower bound is smaller than the NN-distance. It is straightforward to extend the algorithm in order to find the k NN for $k > 1$.

3 Approximation Quality and Cost of NN-Search

In this section, we define two measures for the quality of approximations, namely (1) the mean error of bounds, and (2) the number of vectors visited in the second phase. The objective is to quantify the relation between the granularity of the approximations and approximation quality, as well as the one between the granularity and the search performance. An interesting side issue of the first measure is that the number of bits per dimension of the approximation file grows only logarithmically with d in order to always have the same approximation quality. The second measure, the number of vectors visited in the second phase, allows to predict the response time of VA-File based search given the approximation granularity of the VA-File of an arbitrary data set.

In the following, we characterize a data set by its distance distribution. Let $f(\rho)$ be the distance frequency function, i.e. $f(\rho)$ denotes how frequently there is the distance ρ between two points. The distance distribution $P[dist \leq \rho]$ is the probability that the distance between two points is less than ρ. We further use $f^{lBnd}(\rho)$ to denote the lower bound frequency. Analogously, $P[lBnd \leq \rho]$ denotes the probability that the lower bound is smaller than ρ.

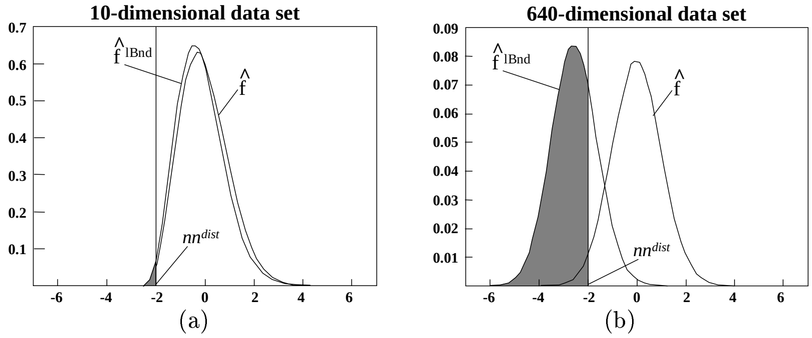


Fig. 2. Frequencies of normalized distances and bounds.

3.1 Deriving the Distribution of the Mean Error of the Bounds

For simplicity, we assume that data points are uniformly and independently distributed in the data space $\Omega = [0,1]^d$, and that each dimension has the same number of bits $\bar{b}$, i.e. $\forall j : b_j = \bar{b}$. Since data is uniformly distributed, the intervals in each dimension are of equal size. Thus, the partition points are $m[j,i] = m[i] = i/2^{\bar{b}}$.

The VA-File uses Equations (2) and (3) to prune the data set. These equations are also valid if we use squared bounds and distances[1]. Therefore, if the squared bounds are close to the squared distances, the real bounds also approximate the distance well. Hence, we define the mean error of the bounds as follows (definition for upper bound is analogous):

$$\varepsilon_l(d,\bar{b}) = \int\limits_{\boldsymbol{p}\in\Omega}\int\limits_{\boldsymbol{q}\in\Omega} \left(dist_2(\boldsymbol{p},\boldsymbol{q})^2 - lBnd_2(\boldsymbol{p},\boldsymbol{q})^2\right) \overrightarrow{dq}\,\overrightarrow{dp} \tag{4}$$

After a number of transformation steps (cf. [16]), we obtain:

$$\varepsilon_l(d,\bar{b}) = d\frac{2^{1+\bar{b}}-1}{6\cdot 2^{2\bar{b}}} \approx \frac{d}{3\cdot 2^{\bar{b}}} \qquad \varepsilon_u(d,\bar{b}) = d\frac{2^{2+\bar{b}}+2^{1+\bar{b}}-1}{12\cdot 2^{3\bar{b}}} \approx \frac{d}{3\cdot 2^{\bar{b}}} \tag{5}$$

Further, the mean squared distance $E(d)$ between points and the standard deviation $\sigma(d)$ in a uniformly distributed data set in the space $[0,1]^d$ are given by: $E(d) = d/6$, $\sigma(d) = \sqrt{7\cdot d/180}$.

To show what this result means, we introduce the notion of a *normalized lower bound distribution*. Let r be a distance. Then, the normalized distance $\hat{r}$ is defined as $\hat{r} = (r - E(d))/\sigma(d)$. In the same way we can normalize the bounds. Let $\hat{f}(\hat{r})$ denote the normalized distance distribution, defined as $\hat{f}(\hat{r}) = f(\hat{r}\cdot\sigma(d) + E(d))$. Analogously, let $\hat{f}^{lBnd}(\hat{r})$ be the normalized lower bound distribution. Using

[1] It is considerably faster to consider only squared distances and bounds to avoid applying the `sqrt`-function.

normalized distances and bounds, it follows that the normalized mean error $\hat{\varepsilon}_l(d,\bar{b})$ is given by $\hat{\varepsilon}_l(d,\bar{b}) = \varepsilon_l(d,\bar{b})/\sigma(d)$.

Observation 1: The smaller $\hat{\varepsilon}_l(d,\bar{b})$, the better the approximations. To show this, we compare two uniformly and independently distributed data sets $\mathcal{D}_A$ and $\mathcal{D}_B$. The first one is low-dimensional, i.e. $d_A = 10$, and the other one is high-dimensional, i.e. $d_B = 640$. The number of bits per dimension is the same for both data sets, i.e. $\bar{b}_A = \bar{b}_B$. Figures 2 (a) and (b) plot the normalized frequencies of $\mathcal{D}_A$ and $\mathcal{D}_B$, respectively. Recall that $\hat{\varepsilon}_l(d,\bar{b})$ is defined as the mean difference between the distance and the bounds, i.e. as the shift between $\hat{f}(\hat{r})$ and $\hat{f}^{lBnd}(\hat{r})$. A vertical line in each figure indicates the normalized k-NN distance. In the figures, we have used a large k, as the effect in Figure 2 (a) is hardly visible when $k = 1$. According to Equation (2), the shaded area corresponds to the relative size of the candidate set. We see that there are much more candidates in the high-dimensional case, and approximation quality is thus much lower. The reason is that the shift $\hat{\varepsilon}_l(d,\bar{b})$ is much larger: from the equations derived above, we see that it grows with $\sqrt{d}$. Consequently, this shift directly reflects the approximation quality. The smaller the shift, the better is the pruning in the first phase and the less candidates are visited in the second phase.

Observation 2: $\bar{b}$ grows logarithmically with d to yield the same approximation quality. To establish an approximation for $\mathcal{D}_B$ with the same quality as one for $\mathcal{D}_A$, $\hat{\varepsilon}_l(d,\bar{b})$ must be equal for both data sets. From that, it follows that the number $\bar{b}_B$ of data set B depends on the one of A as follows:

$$\bar{b}_B = \bar{b}_A + \frac{1}{2} \cdot \log_2\left(\frac{d_B}{d_A}\right) \tag{6}$$

In our example, i.e. $d_A = 10$ and $d_B = 640$, we have to use 3 additional bits per dimension for $\mathcal{D}_B$ to obtain the same approximation quality. Experiments have shown that a good approximation for $d = 10$ needs 5 bits per dimension. Using approximations with $\bar{b} = 12$, the quality of the bounds of a $163{,}840$-dimensional data set is as good as the one of the 10-dimensional data set. We conclude that the VA-File does not degenerate for very high-dimensional data sets.

Observation 3: The error of bounds concentrates around $\varepsilon_l(d,\bar{b})$. Analogously to the mean error of the bounds, one can derive the standard deviations σ_l and σ_u for the error of the lower and the upper bound, respectively:[2]

$$\sigma_l(d,\bar{b}) \approx \frac{\sqrt{d}}{3 \cdot 2^{\bar{b}}}, \qquad \sigma_u(d,\bar{b}) \approx \frac{\sqrt{d}}{3 \cdot 2^{\bar{b}}} \tag{7}$$

Figure 3 (a) shows the frequency of $\varepsilon_l(\boldsymbol{p},\boldsymbol{q})$ for two different numbers of bits per dimension: $\bar{b} = 4$ and $\bar{b} = 6$. The data set was uniformly and independently distributed with $d = 100$ and $N = 100{,}000$. For $\bar{b} = 4$, Equations (5) and (7) yield a mean error of 2.08 and a standard deviation of 0.21. Thus, most errors of the bounds lie around 2. In the case of $\bar{b} = 6$, the bounds are 4 times closer to the real distance on average. A further interesting observation is that for

[2] The derivation is analogously to $\varepsilon(d,\bar{b})$, but is omitted here for lack of space.

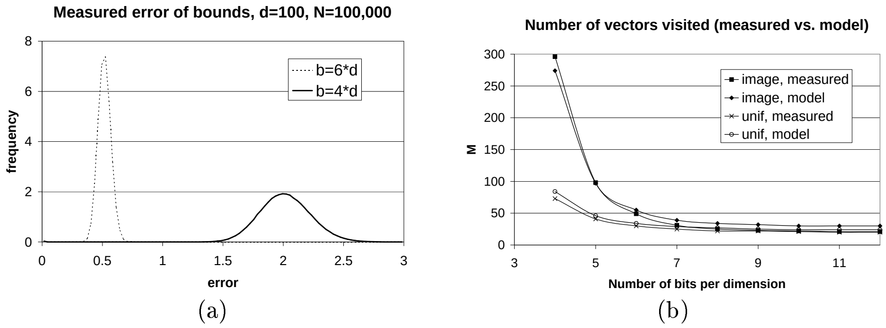


Fig. 3. (a) Frequency of $\varepsilon_l(\boldsymbol{p},\boldsymbol{q})$ for $\bar{b}=4$ and $\bar{b}=6$. (b) Number of visited vectors in a uniform and a real data sets ($d=10$, $N=100,000$).

$\bar{b}=4$, $\varepsilon_l(\boldsymbol{p},\boldsymbol{q})$ is larger than 1.5 in the vast majority of cases. Hence, it is highly probable that $\mathbf{lBnd_2}(\boldsymbol{p},\boldsymbol{q})^{\mathbf{2}}+\mathbf{1.5}$ is also a lower bound on the distance between $\boldsymbol{p}$ and $\boldsymbol{q}$. We will use this observation subsequently to reduce the number of bits per dimension without noticeably losing on result quality.

3.2 Number of Vectors Visited

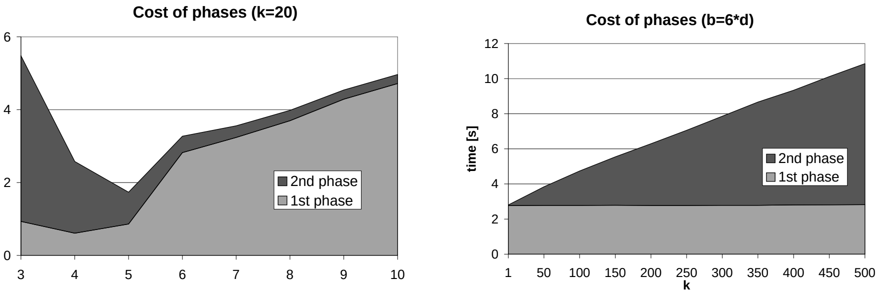
In the previous subsection, we have shown that $\hat{\varepsilon}_l(d,\bar{b})$ reflects the quality of the approximation. However, a measure based on $\hat{\varepsilon}_l(d,\bar{b})$ would not directly tell us the cost of VA-File based search. In this subsection, we determine the number M of vectors visited in the second phase for a given data set and a given k. Note that M is related to $\hat{\varepsilon}_l(d,\bar{b})$: the smaller $\hat{\varepsilon}_l(d,\bar{b})$, the smaller is M. Thus, M also measures the quality of approximation. We derive a formula for M under the assumption that the distance distribution is the same from each point of the data space (*homogeneity assumption* [8]). Note that this assumption is more general than the uniformity assumption in the previous subsection. Thus, our estimation of M applies to real data sets as well.

Expected Nearest Neighbor Distance. A good estimation of the NN-distance of a query is the expected NN-distance of the data set over all possible queries. The probability that the k-NN-distance ρ of an arbitrary query is at most r is:

$$P_k[\rho \le r] = 1 - \sum_{i=0}^{k-1} \binom{N}{i} \left(P[dist \le r]^i (1 - P[dist \le r])^{N-i}\right) \tag{8}$$

We obtain the expected k-NN-distance $\nu(k)$ by integrating over all distances r:

$$\nu(k) = \int_0^\infty r \frac{\partial P_k[\rho \le r]}{\partial r} dr \tag{9}$$



(a) (b)

Fig. 4. The costs of the phases of VA-File based NN-search.

Number of Vectors Visited. The VA-File algorithm visits a point if its lower bound is smaller than the NN-distance. To estimate M, we use the expected NN-distance $\nu(k)$ given by Equation (9):

$$M = N \cdot P[lBnd \leq \nu(k)] = N \cdot \int_{0}^{\nu(k)} f^{lBnd}(x)dx \tag{10}$$

From Equations (9) and (10), it follows that M grows with increasing k. Finally, Figure 3 (b) compares the estimated number for M with the measured one for two data sets and different numbers of bits. The data sets chosen were a 10-dimensional uniformly distributed data set, and a 9-dimensional feature data set of our image database, both with $N = 100,000$. For both cases, one can see that the estimates obtained with Equation (10) are of sufficient quality.

3.3 Cost of VA-File-based NN-search

Given the number of bits per dimension, we now are able to determine the size of the approximation file, and the number M of vectors to visit in the second phase of the search algorithm. Given these two measures, we easily can establish a cost model for VA-File based search. The details are omitted here for lack of space, but are contained in the longer version of this article [16].

Figure 4 depicts the costs of the two phases, obtained experimentally. Figure 4 (a) shows these costs as a function of $\bar{b}$ with $k = 20$, Figure 4 (b) as a function of k with $\bar{b} = 6$. For these figures, we have used a 320-dimensional set of image features with $N = 100,000$. The buffer size was around 25 MB. Consequently, only the approximation files with $\bar{b} < 6$ has fit into main memory. According to the figure, the cost of the first phase varies with $\bar{b}$. But the dependency is not linear due to the limited buffer size and due to the different costs of bit-operations. The second phase depends on both b and k. While the cost of the second phase grows linearly with k, it decreases with growing $\bar{b}$. Figure 4 (a) reveals the optimal setting for $k = 20$: performance is best if $\bar{b} = 5$.

4 Approximating NN-Query Results

In this section, we describe techniques to trade quality for response time for VA-File based search.

4.1 Adapting the Bounds in the First Phase

The evaluation from Subsection 3.1 tells us that $\varepsilon_l(\boldsymbol{p}, \boldsymbol{q})$ is larger than a threshold value α in the vast majority of cases. E.g., for $\bar{b} = 4$, $\alpha = 1.5$, as Figure 3 shows. In other words, $lBnd_2(\boldsymbol{p}, \boldsymbol{q})^2 + 1.5$ is very likely to lower bound the distance, and $uBnd_2(\boldsymbol{p}, \boldsymbol{q})^2 - 1.5$ upper bounds the distance with high probability. Thus we define new bounds as follows:

$$lBnd_2'(\boldsymbol{p}, \boldsymbol{q})^2 = lBnd_2(\boldsymbol{p}, \boldsymbol{q})^2 + \alpha, \qquad uBnd_2'(\boldsymbol{p}, \boldsymbol{q})^2 = uBnd_2(\boldsymbol{p}, \boldsymbol{q})^2 - \alpha \quad (11)$$

Subsequently, we refer to α as *adaptation value*. Given the distance frequency $f^{\varepsilon_l}(\rho)$ for the error of the lower bound, we can determine the probability $P(\alpha)$ that the new bound $lBnd_2'$ is wrong:

$$P(\alpha) \quad = \quad \int_0^{\alpha} f^{\varepsilon_l}(\rho) d\rho \qquad (12)$$

On average, out of the k vectors that are the result of an accurate k-NN search, the adapted bounds of $P(\alpha) \cdot k$ vectors are wrong. In the worst case, these vectors are not in the approximate result. Thus, an upper bound on the average number of false dismissals is given by: $P(\alpha) \cdot k$. Subsequently, we call this approximate query evaluation technique VA-BND.

4.2 Avoiding Vector Visits in the Second Phase

The approximate query evaluation techniques now described reduce the number of vectors visited in the second phase. This is especially important if k is large, as Figure 4 (b) shows. One alternative is to completely omit the second phase, i.e. the vectors with the k smallest lower bounds are the approximate result. We refer to this approximate query evaluation technique as VA-LOW.

In many situations, distances to the query object are meaningless to the user. Rather, a retrieval system assigns scores to the result objects based on a scoring function. With the bounds computed in the first phase and the scoring function, we can approximate these scores. If approximate scores are not sufficient, we have to access the k vectors in the approximate result to determine the exact distances and to compute the scores. With the experiments to follow, VA-LOW-k denotes an approximate query evaluation technique that works like VA-LOW but accesses the k members of the result to determine the exact distance. Note that the results of VA-LOW and VA-LOW-k for a given query contain the same vectors but possibly with a different ranking.

With the above approximate techniques, we may have false dismissals. An upper bound on the number of false dismissals is $M - k$, where M denotes the number of candidates of an accurate search. This is because each of the M candidates may be in the accurate result, but VA-LOW and VA-LOW-k consider only the first k candidates. Consequently, $min(k, M-k)$ upper bounds the number of false dismissals. Since M is typically very small, e.g. $M = 14$ for $k = 10$, the number of false dismissals is also small (4 in the example).

Recall that the VA-File uses the upper bound to reduce the number of candidates according to Equation (3). However, with VA-LOW(-k), the result consists of the k vectors with the smallest lower bound. Thus, computing the upper bounds is not necessary. Finally, note that it does not make sense to combine VA-BND and VA-LOW(-k): VA-BND increases all bounds by α. This does not alter the order of the lower bounds. Since VA-LOW(-k) returns the k vectors with the smallest lower bounds, the result is the same as if we use only VA-LOW(-k).

5 Measuring the Result Quality

In order to quantify the loss of result quality, we introduce two measures: the *ratio of false dismissals over k* and the *normalized rank sum*. Notice that the notion of result quality is different from approximation quality, as introduced in Section 3. While the *approximation quality* describes how good an approximation bounds the real distance to the vector, the *result quality* tells us how close the approximate result is to the accurate result.

Ratio of false dismissals over k. Let $\boldsymbol{r_i}$ be the i-th vector in the accurate result, and $\boldsymbol{x_i}$ be the i-th vector in the approximate result. Further, let $rank(\boldsymbol{x_i})$ be the rank of $\boldsymbol{x_i}$ in the accurate result with $k = N$. Obviously, an approximate result is good if it contains most vectors of the accurate result, i.e. if there are few false dismissals. Our first measure is the ratio of false dismissals rfd:

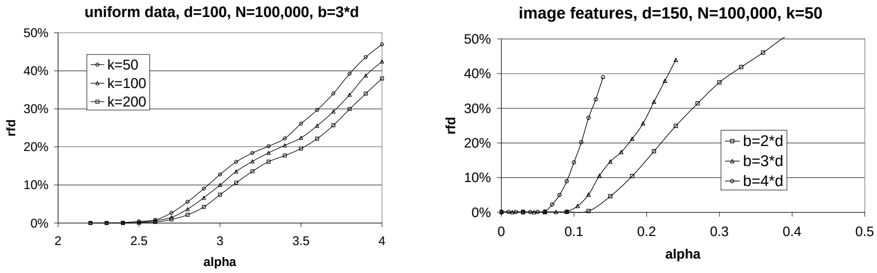
$$rfd \quad = \quad \frac{1}{k} \cdot \sum_{i=1}^{k} \begin{cases} 1 & rank(\boldsymbol{x_i}) > k \\ 0 & otherwise \end{cases} \tag{13}$$

Values of *rfd* close to 0 indicate a high quality of the result.

Normalized rank sum. To illustrate the idea behind our second measure, consider an approximation technique for $k = 10$ which is almost exact. But instead of the 10th best vector, its result contains the 11th best vector, i.e. $\boldsymbol{r_i} = \boldsymbol{x_i}$ for $i < 10$, $\boldsymbol{r_{10}} \neq \boldsymbol{x_{10}}$, and $rank(\boldsymbol{x_{10}}) = 11$. Note that the *rfd*-value does not take into account that the rank of the 10th vector in the approximate result is close to 10. However, it makes a difference whether $rank(\boldsymbol{x_{10}}) = 11$, as in the example, or if $rank(\boldsymbol{x_{10}}) = 20$. In both cases, the *rfd* is the same, but the later result is of lower quality. This consideration leads to a second measure, the so-called *normalized rank sum* (cf. [9]):

$$nrs \quad = \quad \frac{k(k+1)}{2 \cdot \sum_{i=1}^{k} rank(\boldsymbol{x_i})} \tag{14}$$

Values of *nrs* close to 1 indicate a high quality of the result. In the above example, *nrs* is 0.98 with $rank(\boldsymbol{x_{10}}) = 11$, and 0.85 with $rank(\boldsymbol{x_{10}}) = 20$.



(a) (b)

Fig. 5. Result quality *rfd* of VA-BND for (a) a uniformly distributed data set, and for (b) an image feature data set.

6 Evaluation

The evaluation consists of two parts: the first part investigates the result quality, and the second part shows how much we can gain in terms of response time if we allow for a certain ratio of false dismissals. For lack of space, we only consider experiments with quality measure rfd. Further experiments are described in [16].

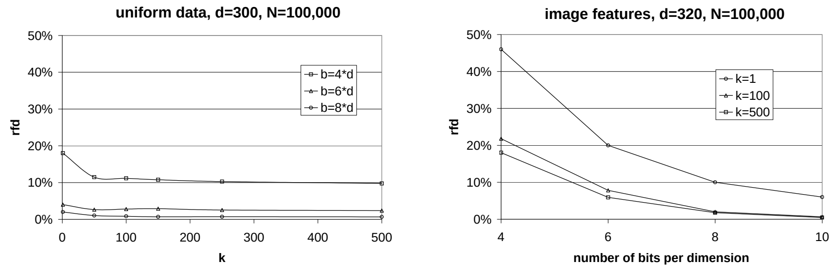
6.1 Result Quality when Approximating the First Phase

In the first experiment, we determined the result quality of VA-BND as a function of k and α ("'alpha"' in the figures). The data set used in Figure 5 (a) consists of 100,000 100-dimensional uniformly distributed points. It depicts the *rfd*-value of an approximate search with $k = 50$, $k = 100$ and $k = 200$ as a function of α. In all experiments, we set $\bar{b} = 3$, i.e. the approximations had a length of $3 \cdot d = 300$ bits. The figure tells us that the number of false dismissals is sufficiently small, i.e. $rfd < 15\%$, for $k > 50$ as long as $\alpha < 3$. An interesting observation is that result quality is better the larger k is. In addition, we can use this figure to determine α based on the quality constraint of the user. For example, if the user specifies the constraint that $rfd < 10\%$ for a 50-NN search with $\bar{b} = 3$, it follows from the figure that α should be smaller than 2.95.

The second experiment shows that VA-BND is also of advantage with real data sets. For Figure 5 (b), we used a feature data set of our image database with $d = 150$ and $N = 100,000$. As with the uniform data set, we can determine α for a given quality constraint based on *rfd*. E.g., for a 50-NN search with $\bar{b} = 2$ and $rfd < 10\%$, we obtain $\alpha = 0.18$.

6.2 Result Quality when Approximating the Second Phase

With the next experiments, we have measured the loss of quality of VA-LOW and VA-LOW-k as a function of k and $\bar{b}$. First, we have used a uniformly distributed data set with $d = 300$ and $N = 100,000$. Figure 6 (a) shows the *rfd*-values of



(a) (b)

Fig. 6. Result quality *rfd* of VA-LOW and VA-LOW-k for (a) a uniformly distributed data set, and for (b) an image feature data set.

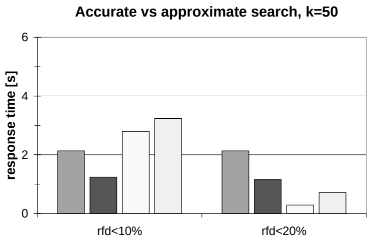
VA-LOW as a function of k for $\bar{b} = 4$, $\bar{b} = 6$, and $\bar{b} = 8$. From this figure, we draw that even for a small number of bits per dimension, the result quality is good. For $\bar{b} = 4$, we see that if k is large, the percentage of false dismissals is around 10%. Only in the case of $k = 1$, one should use more bits per dimension in order to keep *rfd* small. Similarly to determining α, we can use Figure 6 (a) to determine the necessary number of bits to keep result quality above a given value. For example, if the quality constraint is *rfd* $< 10\%$ for a 50-NN search, $\bar{b} = 4$ yields sufficient results. In case of a 1-NN search with the same constraint, we have to use $\bar{b} = 6$ to meet the quality requirement.

Figure 6 (b) shows the *rfd*-values of approximate results with VA-LOW for a set of image features ($d = 320$, $N = 100,000$). The x-axis is the number of bits per dimension. k varies with the different graphs, i.e. $k = 1$, $k = 100$, and $k = 500$. As with uniform data, the more bits we use, the higher the result quality. However, to meet the quality constraint *rfd* $< 10\%$, we need $\bar{b} = 8$ for a 1-NN search, and $\bar{b} = 6$ for $k > 100$.

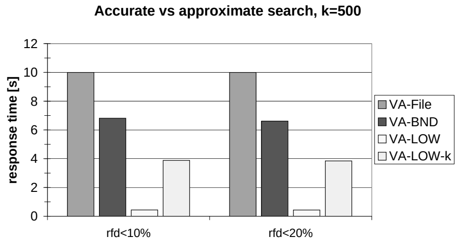
6.3 Trading Quality for Response Time

With the last set of experiments, we want to find out how much we can gain in terms of response time with the approximate search methods compared to accurate search. We have used a data set from our image collections with $d = 320$ and $N = 100,000$. We have varied k and the quality constraint and have measured the response time for each method with its optimal setting ($\bar{b}$, α). The buffer was 25 MB in all experiments. This means that the approximation file has fit into memory if $\bar{b} < 6$. Figure 7 compares the response times of the approximate methods with the ones of an accurate VA-File search (denoted as VA-File) for $k = 50$, and $k = 500$. On the left side of each figure, the quality constraint is $rfd < 10\%$, on the right side it is $rfd < 20\%$. Note that a simple sequential scan through all vectors would last 13 seconds in all settings.

In the case of $k = 50$, VA-LOW and VA-LOW-k are 7 times faster than an accurate search if $rfd < 20\%$, and 50% slower if $rfd < 10\%$ (cf. Figure 7 (a)).



(a)



(b)

Fig. 7. Performance gain of approximate search methods with a quality constraint of $rfd < 10\%$, and $rfd < 20\%$ for (a) $k = 50$, and (b) $k = 500$.

The performance gain in the first case is basically due to lower costs in both phases: in the first phase, no upper bounds are computed, and in the second phase less (or no) vectors are read. Approximate search with VA-BND is 2 times faster than accurate search, and lasts equally long with both quality constraints. Hence, with a rigid quality constraint, VA-BND yields best results, otherwise, VA-LOW is the best choice.

For $k = 500$, we obtain best search performance with VA-LOW and VA-LOW-k: response times are about 20 times smaller than the ones of an accurate search for both quality constraints (cf. Figure 7 (b)). Even for VA-LOW-k, we obtain a performance gain of 60%. In this case, VA-BND yields only a performance gain of 30%.

An interesting observation is that query evaluation times of VA-LOW are smaller for large k than for small k. In our experiments, the search time for $k = 1$ was 2.76 seconds for $rfd < 10\%$, and for $k = 500$ it was 0.44 seconds. The main reason for this is the buffer limitation: VA-LOW needs more bits for $k = 1$ than for $k = 500$ to meet the quality constraint. As a consequence, for $k = 1$, the approximations do not fit into main memory, and, search is considerably slowed down due to IO-operations.

7 Related Work

NN-query evaluation. The vast majority of techniques for exact NN-query evaluation uses trees as underlying index structure [12, 3, 5, 14, 7]. But as the introduction has pointed out, their complexity is linear for large d.

Approximative NN-Search. There are other approaches for approximative NN-search with sublinear complexity even for large d [1, 11], but they come without an evaluation of absolute search times. Furthermore, that complexity typically is the one of the main-memory version of the algorithm, and it is close to the linear one for reasonable approximation errors.

Dimensionality reduction. The principle is as follows: one tries to find a mapping of the vectors to vectors in a space with lower dimensionality that preserves the (ordering of) distances between vectors in the original space as well as possible. A typical dimensionality reduction technique out of many [2] is *Singular Value Decomposition (SVD).* Its results are relatively good, but it is rather expensive. [13] proposes an extension to SVD that operates on aggregates. Performance improves by an order of magnitude, and the loss of quality is acceptable. In this current context, dimensionality reduction is orthogonal to approximate NN-query answering. Dimensionality reduction leads to a more or less exact vector representation of the multimedia objects. Approximate NN-query answering, as investigated in this current article, takes an arbitrary vector representation as starting point.

8 Conclusion

The main contribution of this article are two approximate query-evaluation techniques for the VA-File, a data structure allowing for efficient evaluation of NN-queries in high-dimensional data spaces. To better understand VA-File based query-evaluation techniques, we have derived the distribution of the errors of those bounds. The observation that this distribution is somewhat peculiar has lead to the first approximate evaluation technique. The second technique refers to the second phase of VA-File based search. We have then quantified the loss of quality with these techniques. A 20%-loss of quality, to give an example, typically yields a speedup of 7 (with k=50). For large k, the speedup is even much bigger. Large values of k typically occur with the A0 algorithm that combines ranked results from different subsystems [10].

As 'by-products' from those investigations, this article has contributed the following further results: the approximation quality stays the same if the number of bits per dimension of the approximation file grows logarithmically with the dimensionality of the data space. This means that, from a technical perspective, VA-File based NN-search is feasible for very high dimensional spaces.Furthermore, we have developed a cost model for VA-File based NN-search. Our cost model is useful in many situations, notably for cost-based query optimization.

Acknowledgements. We thank Hans-J. Schek and Martin Breunig for helpful comments.

References

1. Sunil Arya et al. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. Technical report, 1998.
2. D. Barbara, W. DuMouchel, C. Faloutsos, P. J. Haas, J.M. Hellerstein, Y. Ioannidis, H.V. Jagadish, T. Johnson, R. Ng, V. Poosala, K.A. Ross, and K. C. Sevcik. The New Jersey data reduction report. *Data Engineering*, 20(4):3–45, 1997.
3. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, NJ, 23–25 May 1990.

4. S. Berchtold, C. Böhm, B. Braunmüller, D.A. Keim, and H.-P. Kriegel. Fast parallel similarity search in multimedia databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–12, Tucson, USA, 1997.
5. S. Berchtold, D.A. Keim, and H.-P. Kriegel. The X-tree: An index structure for high-dimensional data. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 28–39, 1996.
6. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbour" meaningful? In Catriel Beeri and Peter Buneman, editors, *Proc. 7th Int. Conf. Data Theory, ICDT*, number 1540 in Lecture Notes in Computer Science, LNCS, pages 217–235. Springer-Verlag, 10–12 January 1999.
7. P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, Greece, 1997.
8. Paolo Ciaccia, Marco Patella, and Pavel Zezula. A cost model for similarity queries in metric spaces. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, 1998.
9. A. Dimai. Spatial encoding using differences of global features. In *Storage and Retrieval for Image and Video Databases IV*, volume 3022 of *SPIE Proceedings Series*, pages 352–360, Feb. 1997.
10. Ronald Fagin. Combining fuzzy information from multiple systems. In *Procedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, volume PODS, pages 216–226, Montreal, Canada, June 1996.
11. Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hasing. In *Proceedings of the 25th International Conference on Very Large Data Bases*. Morgan Kaufmann, 1999. Edinburgh, Scotland.
12. A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, MA, June 1984.
13. K. V. R. Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 27(2):166–176, 1998.
14. N. Katayama and S. Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 369–380, Tucson, Arizon USA, 1997.
15. R. Weber, H.-J. Schek, and 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 Databases (VLDB)*, volume 24, New York, USA, August 1998.
16. Roger Weber and Klemens Böhm. Trading quality for time with nearest-neighbor search. Technical report, Dept. of Computer Science, 1999. Available at http://www-dbs.ethz.ch/~weber/paper/EDBT00Long.ps.