Controlling the Precision of Approximate Fast Nearest-Neighbor Searches

Sid-Ahmed Berrani, Laurent Amsaleg, Patrick Gros

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Controlling the Precision of Approximate Fast Nearest-Neighbor Searches

Sid-Ahmed Berrani — Laurent Amsaleg — Patrick Gros

N° 4423

Mars 2002

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--4423--FR+ENG

Controlling the Precision of Approximate Fast Nearest-Neighbor Searches

Sid-Ahmed Berrani* , Laurent Amsaleg † , Patrick Gros †

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Texmex

Rapport de recherche n° 4423 — Mars 2002 — 25 pages

Abstract:

This paper describes a new approach for performing efficient approximated nearest-neighbor searches in high-dimensional databases. This approach allows a fine control over the precision of the search by setting the maximum probability for a vector that would be in the exact answer set to be missing in the approximated set of answers. It assumes that feature vectors are enclosed in clusters. One of the contribution of this paper is the computation of controlled approximations of each cluster. These approximations depend on the distribution of data within each cluster. To answer a query, the search process considers the approximations that correspond to the desired level of precision. This may cause the actual nearest-neighbors of the query point to be ignored. Our method, however, probabilistically bounds the chances for this to happen. This paper also presents a performance study of the implementation. We show, for example, that this method is more than 5 times faster than the sequential scan when it handles more than 1.5 10^6 24-dimensions vectors, even when the probability of missing one of the true nearest-neighbors is below 0.01.

Key-words: Approximate nearest neighbor searche, multidimensional indexing, content-based retrieval

* Thomson Multimédia R&D France, 1, av Belle Fontaine, BP 19. 35511 Cesson-Sévigné - France.
† Équipe Texmex, IRISA-CNRS.

Contrôler la précision des recherches approximatives de plus proches voisins

Résumé : Ce rapport présente une nouvelle technique pour rechercher de façon approchée les plus proches voisins d'un vecteur requête au sein de bases de données hautement multidimensionnelles. Cette technique contrôle la précision de la recherche en fonction de la probabilité maximale que l'on a de ne pas retrouver dans le résultat approché un vecteur qui serait présent dans le résultat exact. Cette technique demande à ce que les vecteurs soient au préalable regroupés dans des partitions (*clusters*). La contribution majeure de ce travail est une méthode pour calculer un ensemble de formes englobantes approchées pour chaque *cluster*. Les caractéristiques des formes englobantes approchées dépendent de la distribution des vecteurs au sein de chaque *cluster* et aussi des niveaux de précision prédéfinis. Conjointement à sa requête, l'utilisateur doit fournir un taux de précision caractérisant la qualité de la réponse qu'il souhaite. Une fois lancée, la recherche n'utilise alors que les formes englobantes approchées qui auront été pré-calculées pour ce taux. Il se peut ainsi qu'un ou plusieurs des plus proches voisins exacts du point requête soient ignorés. Notre méthode borne de façon probabiliste les chances pour que cela arrive. Ce rapport présente également une étude expérimentale des performances de cette technique. Cette étude montre, par exemple, que pour $1,5\ 10^6$ vecteurs ayant 24 dimensions, la recherche approchée est jusqu'à 5 fois plus rapide qu'une recherche séquentielle, même lorsque la probabilité d'ignorer un des plus proches voisins exacts du point requête est inférieure à 0,01.

Mots-clés : Recherche approximative des plus proches voisins, indexation multidimensionnelle, recherche par le contenu

1 Introduction

Image databases and therefore content-based retrieval systems have become increasingly important in many applications areas such as medicine, geography, weather-forecast, molecular biology, art, security, ... Content-based image retrieval systems rely on image processing techniques to extract *descriptors* from images [HKM+97]. Descriptors encode information found in images. They are typically vectors of real numbers defining points in a high-dimensional space. The similarity of two images is assumed to be proportional to the similarity of their descriptors, which is measured as the (typically Euclidean) distance between the points defined by the descriptors. Similarity search is therefore often implemented as a nearest-neighbor search (NN-search) within the feature space.

Nearest-neighbor search is inherently expensive and defining efficient methods for indexing large high-dimensional datasets remains an unsolved problem. All existing indexing schemes generally work well for low-dimensional spaces. Their performance, however, is known to degrade as the number of dimensions of the descriptors increase. This phenomenon is known as the *curse of dimensionality*. In other words, navigating within the index becomes more costly than a simple, brute force, sequential scan in high-dimension spaces when searching nearest-neighbors. However, sequential scans are clearly infeasible for very large image databases.

The curse of dimensionality phenomenon is particularly prevalent when performing *exact* NN-searches. Therefore, there is an increasing interest in performing *approximate* NN-searches, where result quality is traded for reduced query execution time [WB00]. To be useful in practice, this trade-off suggests to make the user able to set himself the precision of the approximate search depending on the desired accuracy. It also suggests that the underlying search process respects faithfully this setting, that is, it must not return an approximate result of a worse quality than the one expected, making in this case the setting meaningless. This is in general very difficult, and most of the existing studies exploring approximate NN-searches do not provide such control, either because of over-simplified assumptions resulting in no real control, or because the knob(s) for setting precision actually control low-level system parameters that, when changed, have very obscure consequences.

In contrast, this paper describes a method that provides an accurate probabilistic control over the precision of approximate NN-searches. This control has the interesting properties to be:

- Clearly understandable by end-users. Users are asked to provide, together with their query, a probability α called the *imprecision level* ($1-\alpha$ is therefore called the precision level). This probability corresponds to the probability for a vector that belongs to the exact answer set to be actually missing in the approximate set of answers that is eventually returned.
- Tunable at run-time. A user can resubmit his query with a different precision setting if he feels that the current setting is not adequate (too coarse for example). Changing the setting will not cause any extra cost to be added on the evaluation of queries. The proposed scheme allows users to gracefully change the precision settings, ranging

from the lossless search, to lower precision levels. As expected, higher precision causes higher response time (much faster than a sequential search, however). In addition, it is possible to use this method to build a system in which the precision of the answer improves as the time goes by.

- All costs are pushed to the off-line phase. The complex calculations of approximate radiuses are all performed off-line. At run-time, once α set, the search algorithm just needs to select the corresponding appropriate radiuses to consider.
- Efficient. Our method is efficient in the sense that it is much faster than the sequential scan and it performance does not severely degrades as the dimension of data and/or the size of the database increase.

The method assumes that vectors are enclosed in minimum bounding hyperspherical clusters. The data points belonging to a cluster are stored together on disk in a sequential manner. Each cluster is analyzed off-line and in addition to the exact radius of each hypersphere, approximate radiuses are computed, each corresponding to a predetermined level of imprecision. The approximate radiuses of a cluster are always smaller than its exact radius.

To answer a query, the user has to provide an imprecision level controlling the quality of the approximate NN-search. This imprecision level drives which approximate radiuses must be taken into account by the filtering rules that eliminate irrelevant clusters. Eliminating clusters on the basis of their approximate radiuses instead on the basis of their exact radiuses may cause the *actual* nearest-neighbors of the query point to be ignored.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 gives a global overview of the search technique. Then, Section 4 presents the method for accurately controlling the precision of approximate NN-searches. Section 5 evaluates the performance of the technique and Section 6 concludes and discusses open issues.

2 Related Work

This section discusses related work. We first overview traditional multidimensional indexing approaches achieving exact NN-searches. We especially focus on the filtering rules theses techniques use to dramatically reduce their response times. We then move to approximate NN-search schemes.

2.1 Cells and Filtering Rules

Traditional multidimensional indexing techniques typically divide the data space into cells containing vectors. Cell construction strategies can be classified in two broad categories: *data-partitioning* and *space-partitioning* indexing methods. Data-partitioning methods such as the X-Tree [BKK96] or the SS-Tree [WJ96] define cells according to the distribution of vectors. Space-partitioning methods such as the LSD$^{\text{h}}$-Tree [Hen98] or [WSB98] divide the data space into cells along predefined lines regardless of the actual distribution of vectors.

NN-algorithms typically use the geometrical properties of cells to eliminate those cells that can not have any impact on the result of the current query [RKV95, HS95]. Eliminating irrelevant cells avoids having to subsequently analyze all the vectors they contain, which, in turn, reduces response time. Tree-based indexing methods typically use this to prune entire branches. Eliminating irrelevant cells is often enforced at run-time by applying two rather similar *filtering rules*. The first rule is applied at the very beginning of the search process and identifies irrelevant cells as follows:

$$\text{if } \mathrm{dmin}(q, C_i) \geqslant \mathrm{dmax}(q, C_j) \text{ then } C_i \text{ is irrelevant,}$$

where $\mathrm{dmin}(q, C_i)$ is the minimum distance between the query point q and the cell C_i and $\mathrm{dmax}(q, C_j)$ the maximum distance between q and cell C_j.[1]

The search process then ranks the remaining cells on their increasing distances to q. It then accesses cells one after the other, fetches all the vectors each cell contains, computes the distance between q and each vector of the cell. This may possibly update the current set of the k best neighbors found so far.

The second filtering rule is applied to abort the search as soon as it is detected that none of the vectors in any remaining cell can possibly impact the current set of neighbors. After having aborted the search, all remaining cells are skipped and the result can be returned. This second filtering rule is:

$$\text{if } \mathrm{dmin}(q, C_i) \geqslant d(q, nn_k) \text{ then abort,}$$

where C_i is the cell to process next, $d(q, nn_k)$ is the distance between q and the current k^{th} nearest-neighbor.

The "curse of dimensionality" problem makes these filtering rules ineffective in high-dimensional spaces. The reasons include high overlap between cells which prevents the rules to filter out many cells and to rapidly abort the search.

2.2 Approximate NN-Searches

The "curse of dimensionality" phenomenon is particularly prevalent when performing *exact* NN-searches. There is therefore an increasing interest in performing *approximate* NN-searches, where result quality is traded for reduced query execution time: approximate NN-searches might miss the true nearest neighbors of query points. Many approaches to approximate NN-searches have been published. We describe below the ones that are the most relevant to our work.

[1] This implicitly assumes that C_j contains at least k vectors. Otherwise, the search process can start applying this filtering rule only once it knows that enough cells have not been filtered out to guaranty the existence of at least k vectors.

2.2.1 Early Stopping Approaches

Weber and Böhm with their approximate version of the VA-File [WB00] and Li et al. with Clindex [LCGMW01] perform approximate NN-searches by aborting the search after having accessed an arbitrary, predetermined and fixed number of cells. These two techniques are efficient in terms of response times, but give no clue on the quality of the result returned to the user and no knob to tune this quality.

2.2.2 Geometrical Approaches

To increase the effectiveness of the above filtering rules, some approaches try to decrease the overlapping rate of cells by considering an approximation of their sizes instead of considering their exact sizes. They typically account for an additional ε value when computing the minimum and maximum distances to cells, making somehow cells "smaller". Smaller cells tend to make the filtering rules more severe, which, in turn, increases the number of irrelevant cells. Cells containing interesting vectors might be filtered out, however. The two above rules become

$$\begin{aligned} \text{if } \mathrm{dmin}(q, C_i) + \varepsilon &\geqslant \mathrm{dmax}(q, C_j) - \varepsilon \\ &\quad \text{then } C_i \text{ is irrelevant;} \\ \text{if } \mathrm{dmin}(q, C_i) + \varepsilon &\geqslant d(q, nn_k) \text{ then abort,} \end{aligned}$$

Determining the appropriate value for ε is key for all approaches using geometrical-based approximations.

In [WB00], Weber and Böhm also present the VA-BND in which ε is empirically estimated by sampling database vectors. They show that this ε is big enough to increase the filtering power of the rules while small enough in the majority of cases to avoid missing the true nearest-neighbors. In addition to be data and query dependent, the main drawback of this approach is that the same ε is applied to all existing cells. This does not account for the possible very different data distributions in cells. While efficient, the VA-BND scheme is not reliable in terms of the precision of the approximate search.

The AC-NN scheme for M-Trees presented in [CP00] (and derived from [AMN+98]) also relies on a single value ε set by the user. Here, ε represents the maximum relative error allowed between the distance from the query point and its exact nearest neighbor and the distance from the query point and its approximate nearest neighbor. In this scheme, ε can only have a very obscure meaning to the user. In addition, their experiments showed that, in general, the actual relative error is always much smaller than ε. Ciaccia and Patella also present an extension to AC-NN called PAC-NN. The PAC-NN scheme uses a probabilistic technique to determine an estimation of the distance between the query vector and its nearest neighbor. It then aborts the search as soon as it founds a vector closer than this estimated distance. The main drawback of PAC-NN is that it assumes known the distribution of data around the query point, which might not be possible in the general case. Also, this scheme can not search for k neighbors.

2.2.3 A Statistical Approach

In contrast to the techniques presented above, DBIN [BFG99] is not a geometrical-based approximation scheme but rather relies on using the statistical properties of data. DBIN clusters data using the EM (Expectation Maximization) algorithm. DBIN aborts the NN-search when the estimated probability for a remaining database vector to be a better neighbor than the ones currently known falls below a predetermined threshold. Unfortunately, DBIN inherits from all the drawbacks of EM such as the sensitivity to the initial settings, the limited number of clusters EM can handle, the assumed Gaussian distribution of data points that vectors may actually not follow or the local optima pitfall. DBIN requires making many strong assumptions which might not hold in practice, making the approximate result very unreliable in the general case.

3 Overview

This section gives a brief overview of our approach to approximate nearest-neighbor searches.

This approach assumes that vectors are enclosed in clusters. The current version of our algorithm for controlling the precision of approximate NN-searches requires clusters to be minimum bounding hyperspheres in an Euclidean space.

All existing vectors might not be in clusters because the clustering algorithm that produced clusters might isolate outliers. If they exist, we assume that outliers are stored in a specific file that we treat separately.

Clusters are analyzed off-line to derive, from the exact radius and the specific characteristics of each, approximate radiuses that will ultimately be considered during the approximate NN-searches. For each cluster, several approximate radiuses are determined, each corresponding to a predetermined level of precision. All the approximate radiuses of one cluster are always smaller than the exact radius of the same cluster. *The main contribution of this paper is the method we use to compute approximate radiuses.* This method will be presented in the next Section.

At query submission time, a user provides, along with the query, a precision level called α controlling the quality of the approximate NN-search. α corresponds to the probability for a vector that belongs to the exact answer set to be actually missing in the approximate set of answers eventually returned.

This precision level then determines which specific approximate radiuses must be taken into account by the filtering rules during the NN-search. Irrelevant clusters are thus filtered out and the remaining clusters are then ranked with respect to the distance of their centers to the query point. Clusters are then accessed one after the other. When a cluster is accessed, all the data points in contains are fetched in memory. The search then computes the distances between all points in the cluster and the query vector. This might in turn update the current set of neighbors. It might also filter out more clusters. The search stops when k neighbors have been found and when the approximate minimum distance to the next cluster is greater than the current distance to the k^{th} neighbor.

Before returning the result to the user, a sequential scan of the file where outliers are stored is performed. This might also update the current set of neighbors.

Our approach enforces approximate NN-searches. It is therefore likely that one or more vectors that are the true nearest-neighbors of a query point are missed, and are consequently not in the approximate answer set. Vectors are missed because the clusters in which they are kept are filtered out. Since clusters group vectors that are quite similar, it is therefore possible that most of the nearest-neighbors of a query point are in a unique cluster. If this cluster is filtered out, then most of the true nearest neighbors of a query point are missed. When searching for k neighbors, while the probability of missing one true neighbor is α, the probability of missing all the k nearest neighbors is of the order of α too.

4 Computing Approximate Radiuses and Controlling Precision

This section presents the method that computes the approximate radiuses of each clusters. This method takes into account all the diversity of the characteristics of clusters such as the distribution of the vectors they contain, their sizes, their population. It computes these approximate radiuses for several levels of imprecision. Given the level of imprecision actually set by the user at run-time, the system selects for each cluster the corresponding appropriate radius to consider during the search.

Since approximations of clustered are considered, it is possible that the filtering rules filter out clusters in which some of the true nearest-neighbors of the query point are. It is therefore possible that one or more vectors that would be in the exact answer set are actually missing in the approximate set of answers eventually returned to the user. Such vectors are called *missing vectors*.

If we focus on one specific missing vector, because the approximate NN-search considers approximate radiuses instead of exact (and bounding) radiuses, then we can assert this vector is located in a very particular area of the specific cluster in which this vector belongs, area bounded between the exact and one particular approximate radius of that cluster.

This section presents how the volume of this area for every single cluster is controlled, since, for one cluster, this volume is linked to the probability for a vector of that cluster, if it is in the exact result, to be missing in the approximate result. For the sake of clarity, the method is described here for a single imprecision level called α. Extension to several levels is straighforward.

The presentation of the technique is decomposed in 4 steps:

1. We first explain where missing vectors can only be located in each cluster, and why the volume of the area in which they can only be is directly determined by the imprecision of the search.

2. We then explain the relation between the global level of imprecision α and the local imprecision settings α_i that have to be considered at the level of each cluster.

3. We then detail how we use each α_i to compute the corresponding approximate radius of each cluster if we assume that vectors in clusters follow an isotropic distribution.

4. We finally relax this assumption and present a more general way to compute approximate radiuses, even if the distribution of vectors in clusters is severely skewed.

4.1 Hyper-rings and Hyper-caps

This first part explains where missing vectors can only be located in each cluster. We focus on one specific cluster, C_i. The minimum hypersphere bounding this cluster, called MHS_i, is centered on c_i and its radius is r_i. We also suppose that one approximation of this cluster has been computed somehow. This approximation is an hypersphere centered on c_i with a radius equal to $r_i - \varepsilon_i$. This approximate hypersphere is called AHS_i. C_i contains several vectors, but one specific vector, called V, is missing in the set of answers returned by the approximate NN-search while it was included in the set of answers returned by the exact search. V is therefore a missing vector.

Because V is a missing vector, then the three following lemmas can be asserted:

Lemma 1. C_i in which V is has been filtered out by the filtering rules. Otherwise, if C_i had not been filtered out, then all vectors from C_i would have been considered by the approximate NN-search, and therefore V would have been discovered and would not be missing. (C_i is obviously not filtered out in the case of the exact search since one of its vector, namely V, is in the answer set.)

Lemma 2. V is inevitably located in the very specific area of C_i which is upper-bounded by MHS_i and lower-bounded by AHS_i. The distance between V and c_i can not be greater than r_i, otherwise V would not belong to C_i. The distance between V and c_i can not be less than $r_i - \varepsilon_i$ otherwise V would never have been missing in the set of approximate answers. The area in which V can only be is called the *hyper-ring* of C_i, noted HR_i.

Lemma 3. If dmin_i denotes the minimum distance between a query point q and MHS_i, then V in C_i is *precisely located* in the region of C_i enclosed between the intersection of MHS_i and the hypersphere centered on the query point q with a radius equal to $\mathrm{dmin}_i + \varepsilon_i$. If V was further away from q, then V could not be missing in the approximate result if it was in the exact result because in this case the cluster would not have been filtered out. This specific region is included in HR_i and is called the *hyper-cap* of cluster C_i, noted HC_i.

Figure 1 provides a graphical representation of Lemmas 2 and 3, and shows what are an hyper-ring and an hyper-cap, in a 2D, Euclidean space, however. This Figure shows a unique cluster C_1, characterized by its center (not labeled), its radius (labeled as r_1) defining its minimum bounding hypersphere MHS_i. The approximate hypersphere AHS_i of C_1 is depicted as the grey dashed circle on the Figure. Note the radius of this approximate hypersphere is $r_1 - \varepsilon_1$. The area of C_1 that is external to this approximate hypersphere while

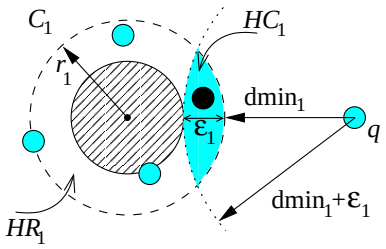

Figure 1: One Hyper-ring and one Hyper-cap

still enclosed in C_1's minimum bounding hypersphere is the hyper-ring of cluster C_1, noted HR_1 on the Figure. C_1 contains 4 vectors, 3 represented by grey dots, and another vector represented by a solid black dot. This later vector is V. In addition, Figure 1 also shows a query point q and the minimum distance from q to C_1 (dmin_1 on the Figure). V is a missing vector, therefore it can only be in HR_1. Moreover, V can only be in the specific area of HR_1 defined by the intersection of the hypersphere bounding C_1 and the hypersphere centered on q that has a radius equal to $\text{dmin}_1 + \varepsilon_1$ and depicted as a plain grey area on Figure 1. This area corresponds to the hyper-cap of C_1, called HC_1.

It is quite obvious that the volume of the hyper-ring of one cluster is related to the imprecision of the search (hence to ε_i) because the larger the volume of the ring of one cluster, the higher the probability for any vector of that cluster to be located within that ring (and possibly within the cap once the query point is known). The next section investigates the relationships between HR_i, HC_i and α.

4.2 From α to α_i

The previous section defined hyper-rings, hyper-caps and pointed out the relationships between the volume of rings and the imprecision of the search. This section is a first step towards explaining how approximate radiuses corresponding to a given α are determined. This first step presents how we relate α to probabilities at the level of each cluster, which will ultimately be at the root of the computation of approximate radiuses.

Enforcing the *global* level of imprecision α asks to control, at the level of each cluster, the *local* probability α_i for a vector of that cluster, if it is in the exact result, to be missing in the approximate result.

Let us introduce the following notations:

- $P(\text{miss}(V))$ is the probability for a vector V that belongs to the result returned by an exact NN-search to be missing in the set of answers returned by the approximate NN-search; It must verify $P(\text{miss}(V)) \leqslant \alpha$. This is the global imprecision constraint.
- $P(V \in C_i)$ is the probability for a vector V to be located within cluster C_i.

- $P(\text{miss}(V)|V\in C_i)$ is the probability for a vector V that belongs to the result returned by an exact NN-search to be missing in the set of answers returned by the approximate NN-search, knowing that vector is located within cluster C_i. Our goal is to find α_i such that, $\forall i, P(\text{miss}(V)|V \in C_i) \leqslant \alpha_i$ enforces the global imprecision constraint.

Because each vector belongs to one and only one cluster, $(V \in C_i)$ forms a partition of the event space. Hence, $P(miss(V))$ can be formulated as follows:

$$P(\text{miss}(V)) = \sum_j P(\text{miss}(V)|V\in C_i)P(V\in C_i) \tag{1}$$

From this equation, we are trying to determine $P(\text{miss}(V)|V \in C_i)$, for every single cluster C_i. Our goal is therefore to solve equation (1) and to find what are the probabilities α_i local to each cluster C_i induced by having set α.

Without any a priori knowledge on the distribution of data and/or on the possible distribution of the query vectors, it is impossible to estimate $P(V \in C_i)$ for each V and each C_i, and therefore impossible to compute α_i, unless we consider that $P(\text{miss}(V)|V \in C_i) = P(\text{miss}(V)|V\in C_j), \forall i, \forall j$. Considering this means that the probability of missing a vector is uniformly distributed over all clusters, independently of their relative position to the query point or to their population. We are well aware that a careful analysis of the data space could possibly give better hints for estimating $P(V \in C_i)$ and $P(\text{miss}(V)|V \in C_i)$, possibly enforcing more precise approximations. This is, however, an open problem that we may study in the future.

Therefore, by denoting $\beta = P(\text{miss}(V)|V\in C_i)$, it is possible to rewrite equation (1) as:

$$\begin{aligned} P(\text{miss}(V)) &= \sum_j \beta P(V\in C_i) \leqslant \alpha \\ P(\text{miss}(V)) &= \beta \sum_j P(V\in C_i) \leqslant \alpha \\ P(\text{miss}(V)) &= \beta \leqslant \alpha \end{aligned} \tag{2}$$

From (2), it is possible to conclude that

$$P(\text{miss}(V)|V\in C_i) \leqslant \alpha_i \leqslant \alpha, \ \forall i \tag{3}$$

that is, enforcing $P(\text{miss}(V)|V\in C_i) \leqslant \alpha$ for each cluster C_i ensures that $P(miss(V)) \leqslant \alpha$. The next section shows the radiuses of approximate hyperspheres given the imprecision level α are computed.

4.3 From Hyper-rings, Hyper-Caps and α to Approximate Radiuses

Sections 4.1 defined hyper-rings, hyper-caps and outlined the relationships between the volume of rings and the imprecision of the search. Section 4.2 defined what is the probability

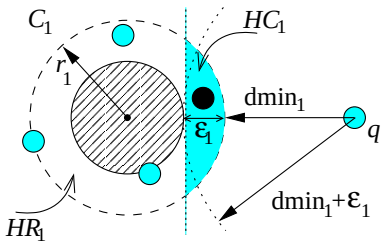

Figure 2: Using an Hyperplan to Approximate HC_1

for a vector belonging to a specific cluster to be missing if it is returned by the exact search. This section is the last step towards explaining how approximate radiuses are determined given α. To reach this goal, we first present how the number of vectors that are in the hyper-ring of each cluster is determined. We then explain that, under the specific assumption of isotropy, the number of vectors that are in the hyper-cap of each cluster is a simple proportion of what is in the hyper-ring. We will then be ready to present the numerical method used to compute radiuses.

To enforce inequation (3), the number of vectors that are located in the hyper-cap of cluster C_i (noted card(HC_i)) over the total number of vectors belonging to C_i (noted card(C_i)) must be less or equal to α, that is:

$$\frac{\text{card}(HC_i)}{\text{card}(C_i)} \leqslant \alpha, \tag{4}$$

Inequation (4) indicates that, to enforce α at the level of cluster C_i, the approximate radius that determines the volume of HR_i must be such that the resulting hyper-cap HC_i contains at most the proportion α of all the vectors of C_i.

card(HC_i) depends on the distance between q and cluster C_i because this impacts the (inner) curvature of HC_i. For a cluster C_i and a given approximate radius, the further q, the bigger the hyper-cap of cluster C_i. Depending on the distance between q and each cluster is very problematic since it is not possible to account for all possible locations of q. It is possible to break this dependency by considering an overestimation of HC_i, called $\widehat{HC_i}$, defined as the intersection of C_i with an half-subspace. This half-subspace contains q and is bounded by an hyperplan orthogonal to (q, c_i) and located at a distance of $\text{dmin}_i + \varepsilon_i$ from q. Note that one hyper-cap defined using such an hyperplan *is always larger* than the same hyper-cap defined using its actual curvature. Relying on hyperplans is therefore a safe approximation since it overestimates the probability to miss a vector. Figure 2 shows an example of this hyperplan, denoted as a dotted straight line. Note the volume of the plain grey area increased with respect to Figure 1. From now on, all hyper-caps mentioned in the rest of this paper must be understood as approximate by hyperplans.

card($\widehat{HC_i}$) also depends on the distribution of the data points within HR_i (and therefore within C_i) and also on which "side" of this cluster the query point is close to. If the

distribution of the data points within cluster C_i follows a privileged direction, $\text{card}(\widehat{HC}_i)$ will be very different if q is also aligned with this direction (in this case $\text{card}(\widehat{HC}_i)$ will be large), or if q is orthogonal to this direction (in this case $\text{card}(\widehat{HC}_i)$ will be small). For simplicity, we assume in the rest of this section that points in each HR_i follow an *isotropic distribution*.[2] Section 4.4, however, reconsiders this assumption and accounts for more realistic distribution models.

Because points in all hyper-rings are supposed to follow an isotropic distribution, the number of vector in one hyper-cap is a proportion of the number of vectors in the corresponding hyper-ring. Therefore, $\text{card}(\widehat{HC}_i)$ can be expressed as:

$$\begin{aligned}\text{card}(\widehat{HC}_i) &= \text{card}(HR_i)\frac{\text{vol}(\widehat{HC}_i)}{\text{vol}(HR_i)} \\ &= \text{card}(HR_i)\frac{\text{vol}(\widehat{HC}_i)}{\text{vol}(MHS_i)-\text{vol}(AHS_i)} \end{aligned} \quad (5)$$

where $\text{vol}(\widehat{HC}_i)$ is the volume of the hyper-cap of C_i, $\text{vol}(HR_i)$ is the volume of the hyper-ring of C_i, $\text{vol}(MHS_i)$ the volume of the minimum hypersphere bounding C_i and $\text{vol}(AHS_i)$ the volume of C_i's approximate hypersphere.

Using equation (5), it is possible to rewrite inequation (4) as follows:

$$\frac{\text{vol}(\widehat{HC}_i)}{\text{vol}(MHS_i)-\text{vol}(AHS_i)}\frac{\text{card}(HR_i)}{\text{card}(C_i)} \leqslant \alpha \quad (6)$$

Except $\text{vol}(MHS_i)$, all the terms in inequation (6) depend on the size of the approximate hypersphere of C_i. The radius of this approximate hypersphere is the unknown to find from this unique equation. We solve inequation (6) by first making the following preliminary remarks:

- $\text{card}(HR_i)$ can not be expressed analytically.
- $\text{card}(HR_i)$ decreases when the radius of C_i's approximate hypersphere increases.
- $\text{card}(HR_i)$ is not continuous and varies within $\mathbb{N}$.

Given these remarks, it is possible to resolve *numerically* inequation (6) and to obtain the value for the radius of the approximate hypersphere of C_i given one specific α. The numerical method we use relies on a bisector-based strategy within the interval $[0, r_i]$. Due to lack of space, we do not include here the corresponding pseudo-code.

[2] We do not assume, however, that all hyper-rings of all clusters are identical. Their distribution only share the isotropic property.

4.4 Arbitrary Distributions in HR_i

The previous section presented the approach used to compute approximate hyperspheres when the vectors in the hyper-rings follow an isotropic distribution. This assumption does clearly not hold in the general case. Therefore, it is possible to extend the approach to cope with arbitrary distributions of vectors within hyper-rings on a per cluster basis, that is, the algorithm adapts its behavior to the specific distribution encountered in every single cluster.

The previous section assumed an extreme case in the sense that it assumed the isotropic distribution to hold for all HR_i. Another extreme case is to consider that the vectors in all the hyper-rings of all existing clusters *are all* included in their associated hyper-cap. In this case, the vector distribution is severely skewed and the isotropic assumption does not hold anymore.

In fact, for a given α, the real distribution of vectors in hyper-rings is, for each cluster, between the two above extreme cases. We can therefore try to estimate, for each cluster, the probability for the isotropic distribution to hold. For cluster C_i, this probability is called $P(ID_i)$. This probability allows to rewrite $P(\text{miss}(V))$ as:

$$\begin{aligned} P(\text{miss}(V)) = & P(\text{miss}(V)|ID_i)P(ID_i) + \\ & P(\text{miss}(V)|\overline{ID}_i)P(\overline{ID}_i) \end{aligned} \tag{7}$$

where $P(\text{miss}(V)|ID_i)$ is the probability of having V missing when the cluster is isotropic. Supposing this assumption holds for C_i, then $P(\text{miss}(V)|ID_i)$ can be deduced from inequation (6). If it it does not hold, then, from (4), $P(\text{miss}(V)|\overline{ID}_i) \leqslant \frac{\text{card}(HR_i)}{\text{card}(C_i)}$. It is therefore possible to rewrite (7) as:

$$\begin{aligned} P(\text{miss}(V)) = & \\ & \left(\frac{\text{vol}(\widehat{HC}_i)}{\text{vol}(MHS_i) - \text{vol}(AHS_i)} \frac{\text{card}(HR_i)}{\text{card}(C_i)} \right) P(ID_i) + \\ & \left(\frac{\text{card}(HR_i)}{\text{card}(C_i)} \right) P(\overline{ID}_i) \leqslant \alpha \end{aligned}$$

In this inequality, $P(ID_i)$ is the only unknown. In general, determining what is the right value for this probability for each cluster require to use statistical approaches. For one cluster C_i and given one predetermined α, it is possible to estimate $P(ID_i)$ by first computing C_i's approximate hypersphere under the assumption of having an isotropic distribution, and then estimate how many vectors fall in C_i's hyper-cap. Then, several random vectors can be generated and used as queries. It is thus possible to count how many vectors of C_i miss in the approximate result. Comparing the estimated number of missing vectors to the number of vectors that actually miss is key to tune $P(ID_i)$.

4.5 Wrapping-up

This section presented how the approximated radiuses of clusters are computed in order to enforce controlled approximated NN-searches. Approximated NN-searches may potentially miss the true nearest neighbors of one query point. We therefore first explained where missed vectors can only be in each approximated cluster. We then explained that the desired imprecision level of the search has to determine the volume of the areas in which missing vectors can only be. We then gave the equations to numerically solve in order to compute the value of the approximated radius of each cluster. This computation accounts for arbitrary distribution of vectors within each clusters. Coping with various levels of imprecision is possible by simply repeating the above calculations, each time using a different predetermined α.

It is important to note that, in this Section, α is an upper bound on the probability for one vector to be missed by an approximate NN-search. When searching for k nearest neighbors, the probability to miss all of them is not α^k, because the different miss(V_i) are *not independent events*. As a matter of fact, because the clustering tends to group together very similar vectors, it is possible that most of the nearest-neighbors of a query point belong to the same cluster, if not located in a small area of one cluster, making the event of missing one not independent of the event of missing all of them. Therefore, the probability of missing all the vectors is closer to α than it is to α^k.

5 Performance Evaluation

This section provides an performance evaluation of the technique detailed above. The first experiment shows how evolves the precision of the approximate results. The second experiment shows the impact of the dimensionality of data over the response time. Finally, the third and last experiment shows the performance of the search technique when the size of the database increases. We first describe our experimental setup.

5.1 Experimental Environment

The technique computing approximate radiuses and the corresponding NN-search algorithm are implemented in C++. All the algorithms run on a Sun Blade 100 workstation running SunOS 5.7. Its CPU is a 502 MHZ UltraSPARC-IIe, with 763 Mb of main memory and 73 Gb of local secondary storage. All the response times reported here have been obtained using `getrusage()`.

5.2 Implementation of the Clustering

As pointed out Section 4, our approach requires that feature vectors are clustered in minimum bounding spheres. The clustering algorithm we use is derived from the first phase of Birch [ZRL96]. It has couple crucial differences, however. Birch ends its first phase when all the created micro-clusters can fit in the allowed main memory. Instead, we stop

our clustering when the number of micro-clusters created falls bellow the maximum number of clusters that are allowed to exist. The variance of data points drives the radius of Birch' clusters. Instead, radiuses of clusters in our implementation are exact in the sense that each defines a minimum bounding hypersphere. Birch uses a hierarchical approach to quickly find the potential clusters in which one point might fall, which therefore avoids the need to examine many irrelevant clusters. This hierarchical approach greatly accelerates the insertion of points in clusters, but may also cause many points to be misplaced. Our approach for clustering does not use any (accelerating) hierarchy. Instead, it examines all existing clusters every time one point has to be inserted. We are well aware that this is not smart, that it does not scale and that it makes the clustering awfully time-consuming. We have to point-out, however, that we were primarily interested in controlling the precision of approximate NN-searches. We therefore tried to be as less as possible perturbed by bogus assignments of points in clusters. Not using any hierarchy tends to strongly diminish point mis-placements. Coming up with a smart clustering strategy is part of our ongoing work.

The output of the clustering phase is a set of minimum bounding hyperspheres defined by their center and their exact radius. As for Birch, clusters might overlap and outliers are treated separately. Data points are stored sequentially on disks on a per cluster basis. No specific data structure is used to index the clusters. Outliers are also stored in a separate data file, in a sequential manner.

5.3 Overview of the Database

To evaluate our approach, we used a 143 Mb DB made of the 24 dimension descriptors derived from 13,548 images. 610 images come from a DB of still images found on the web[3], while all other images come from various TV broadcasts. The descriptors computed belong to the local differential descriptors family [FRKV94, SM97], extended to cope with color [AG01, GFB01]. On average, more than 110 descriptors are computed for each image. Therefore, the total number of descriptors for our image bank is 1,499,869. Among these images, we know particularly well 1,816 images (413,412 descriptors) because we extensively used them in past works [AG01, AGB01].

5.4 Experiment 1: Precision of the Search

The goal of this experiment is to study the precision of the result returned by an approximate NN-search, with respect to the probability α.

For this experiment, we focused solely on the well known 1,816 images (413,412 descriptors, 24D). The noise rate for the clustering was set to 0.20, that is, clusters containing less than 20% of the average population of all clusters are considered to contain noise. The corresponding vectors are the outliers that we treat separately. Given this DB and this noise rate, the clustering algorithm created 727 clusters and found 101,971 outliers. We also

[3] http://www.inrialpes.fr/movi/pub/Images/index.html

created a set of 200 queries. The vectors used as queries were randomly picked in the DB. Each query asks for 15 neighbors.

To study the precision of approximate NN-searches, we first retrieved the 15 exact NN of each query vector. We then performed the 200 approximate NN-searches varying α. Then, for each result set, we computed the proportion of the exact answers that were not retrieved by the approximate process. We then averaged all these proportions. This averaged proportion corresponds to an empirical estimation of α, and reflects the overall imprecision of the search. Figure 3 presents these results for various values of α and for various probabilities $P(ID)$.[4]

This Figure shows the average proportion of missing vectors given the probability α set. It shows, for exemple, that when α is set to 0.3, that is, the approximate search is allowed to miss 5 (among 15) of the exact nearest neighbors of each query point, then, when $P(ID) = 1$, the average proportion of missing vector observed in all the 200 approximate results sets is 0.27, that is, on average, the approximate NN-searches misses 4.05 vectors that are the actual NN of the query points. For all $P(ID)$, the measured imprecision is always smaller than α (no points are above the diagonal line).

We must point out, however, as mentionned Section 4.5, that a side effect of the clustering of points is that it is possible to miss all the true neighbors because they are all in a unique cluster. Therefore, it is possible that, in some cases, the effective precision is 0. This actually happened 7 times for $\alpha = 0.3$ and $P(ID) = 1$.

Varying $P(ID)$ has subtle effects. Recall this gives the probability for the isotropic distribution to hold. When $P(ID) = 0$, the approach assumes that all vectors in hyper-rings are all in hyper-caps. This tends to make the corresponding approximate radiuses (almost irrespective of α) extremely close to the exact radius of the minimum bounding hypersphere of all clusters. This filters out many clusters, but not enough to possibly miss vectors. This, in turn, causes the actual precision of the search to be much higher than the one set. This can be observed on the Figure, where, for small $P(ID)$, the actual number of missing vectors is also very small. In contrast, when $P(ID)$ is close to 1, then approximate radiuses are small enough (with respect to the radiuses of the minimum bounding spheres of clusters) to restore the effectiveness of the filtering rules. Since many clusters are filtered out, it is more likely that the true NN of query points are missed. This is why, for example, the line showing the imprecision when $P(ID) = 1$ follows closely the diagonal line. As a matter of fact, it seems that our clusters are on average quite isotropic.

5.5 Experiment 2: Influence of Data Dimensionality

The second experiment shows the influence of the dimensionality of data on the performance of the search. In this experiment, we re-used again the previous 413,412 descriptors having 24 dimensions. These descriptors were then truncated to 3, 5, 7, 10, 15, 20 dimensions. Each resulting dataset was then clustered (same noise ratio as the one used previously). Table

[4] $P(ID)$ means here that the probability $P(ID_i)$ is the same for all the clusters.

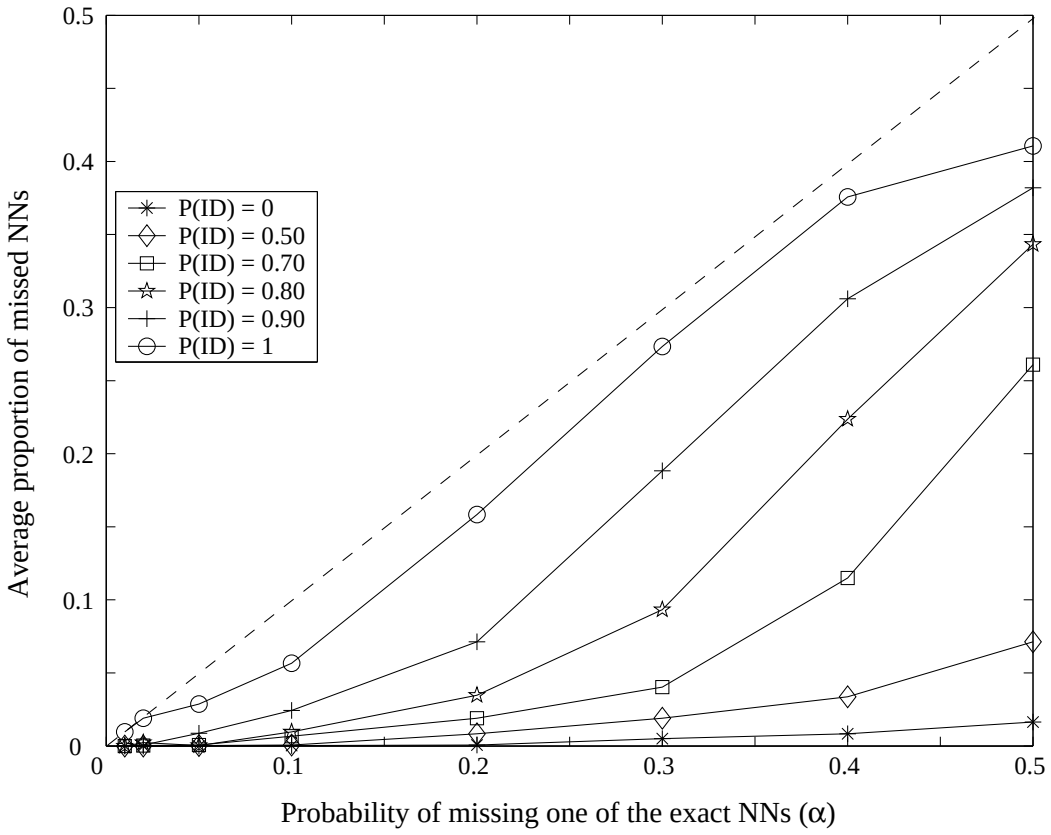

Figure 3: 413,412 desc., 24d, 200 q. vectors, 15 NNs, estimating the probability of missing an exact NN in the approximate result.

1 summarizes the characteristics of these datasets.[5] We also reused the previous set of 200 query vectors (truncated accordingly). In this experiment, $P(ID) = 1$.

In this experiment, we gave 4 different values to α. One of them is 0. In this case, all the approximate radiuses considered during the search are equals to the exact radiuses of all minimim bounding spheres. Therefore, $\alpha = 0$ gives the exact answer. Irrelevant clusters that are filtered out can not, in this case, contain any missing vector. Figure 4 shows the resulting cumulative response time when searching the 15 NN of each of the 200 query

[5] In this table, Max pop and Min pop correspond to the maximum and minimum number of vectors found in the clusters.

Dim.	# of clust.	# of outliers	Max pop	Min pop
3	912	45,703	5,526	44
5	972	48,717	10,616	38
7	905	52,286	17,528	43
10	856	61,238	13,858	42
15	969	65,437	13,311	43
20	709	69,991	21,565	50
24	727	101,971	12,063	56

Table 1: Description of the databases for various dimensions.

vectors, varying the dimension of the vectors. This Figure also shows the response time of a traditional sequential scan since it remains competitive in high-dimensions.

As it can be observed, the performance of our approach is linear, while much below the sequential scan. In the case of 20 dimensions, a sequential scan takes 98,87 seconds. In this experiment, 709 clusters were created, 343,421 vectors were clustered and there was 69,991 outliers in the case of 20 dimensions. In this case, the exact search (i.e., $\alpha = 0$) with our method took 48.75 seconds, that can be decomposed in 38.87 seconds for the cluster-based search and a little less than 10 seconds for sequentially scanning the outliers. In this case, 526 clusters, on average, were filtered out. For $\alpha = 0.01$, in 20 dimensions, the overall response time is 18.43 seconds, decomposed in 9.85 seconds for the cluster-based approximate search and a little less then 9 seconds for scanning the outliers. In this case, 688 clusters, on average, were filtered out. Considering only 3% of the cluster, in turn, eliminated more than 91% of the vector from the analysis.

It is interesting to see on this Figure that no major response time improvements are observed even when α is increased (and therefore the precision decreased). The reason is that –if we consider only one cluster for simplicity– when α is increased, then the relative differences between consecutive radiuses tend to be smaller since the density of the cluster increases (its volume decreases much faster than the number of points in the approximate hyperspheres), which, in turn, does not allow the filtering rules to filter out much more clusters.

5.6 Experiment 3: Varying the DB Size

The last experiment we show here uses the full database described above, that is, the 1,499,869 descriptors of 24 dimensions computed on the 13,548 images. To vary the size of the DB, however, we generated various data sets by keeping only 50,000, 100,000, 200,000, 300,000, 413,412, 500,000, 750,000, 1,000,000 and 1,250,000 of them. Each dataset was then clustered (noise rate=0.2) and outliers stored separately. For exemple, 850 clusters were created and 147,775 outliers were found in the case of the dataset containing 500,000 descriptors. We used again our 200 query vector set. Figure 5 shows the cumulative response times for the search of 15 NNs for each query vector, for 4 values of α and for the sequential scan. In this experiment, $P(ID) = 1$.

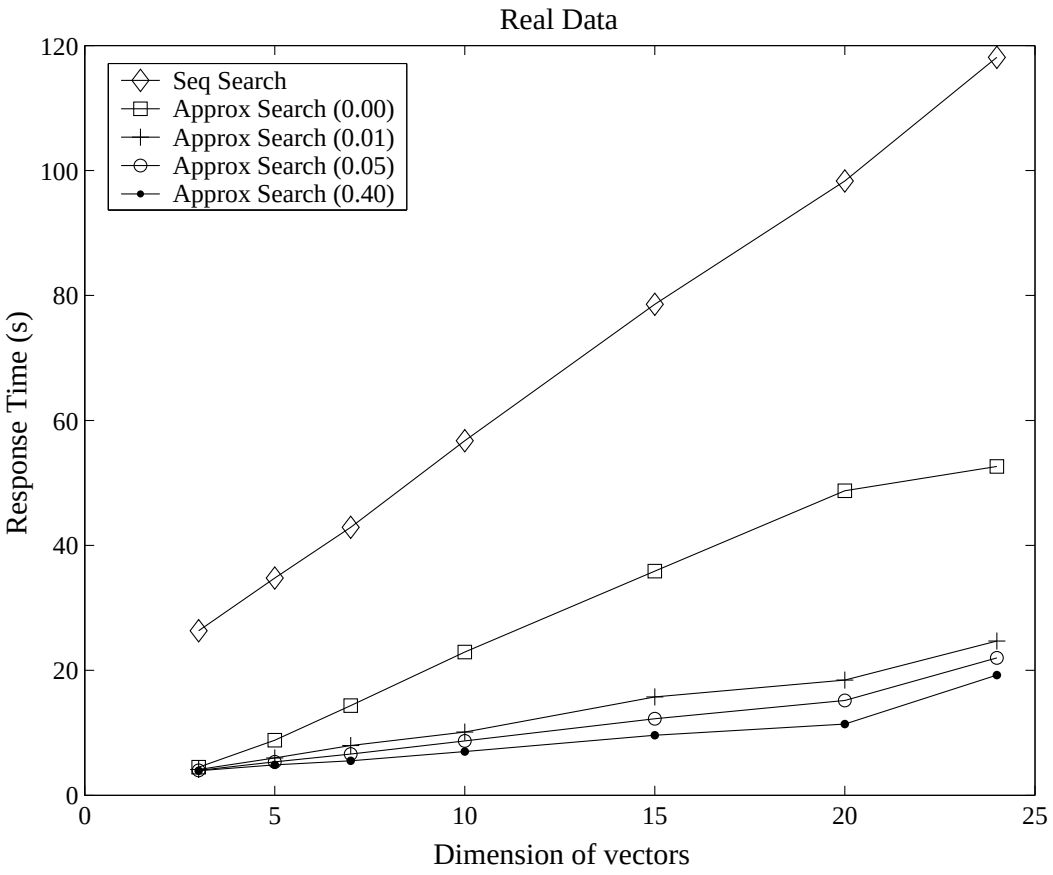

Figure 4: 413,412 desc., 200 q. vectors, 15 NNs, increasing dimension of vectors.

For the database of 1,499,869 vectors, the response time of the cluster based search with $\alpha = 0$ is half the response time of the sequential scan. With a α equals to 0.01, our technique searches the 15 approximate neighbors in about 5 times faster than the sequential scan, while the average effective precision stills greater than 0.99. In this case, on average, 97.77% clusters were filtered out corresponding to 92,87% of the clustered vectors.

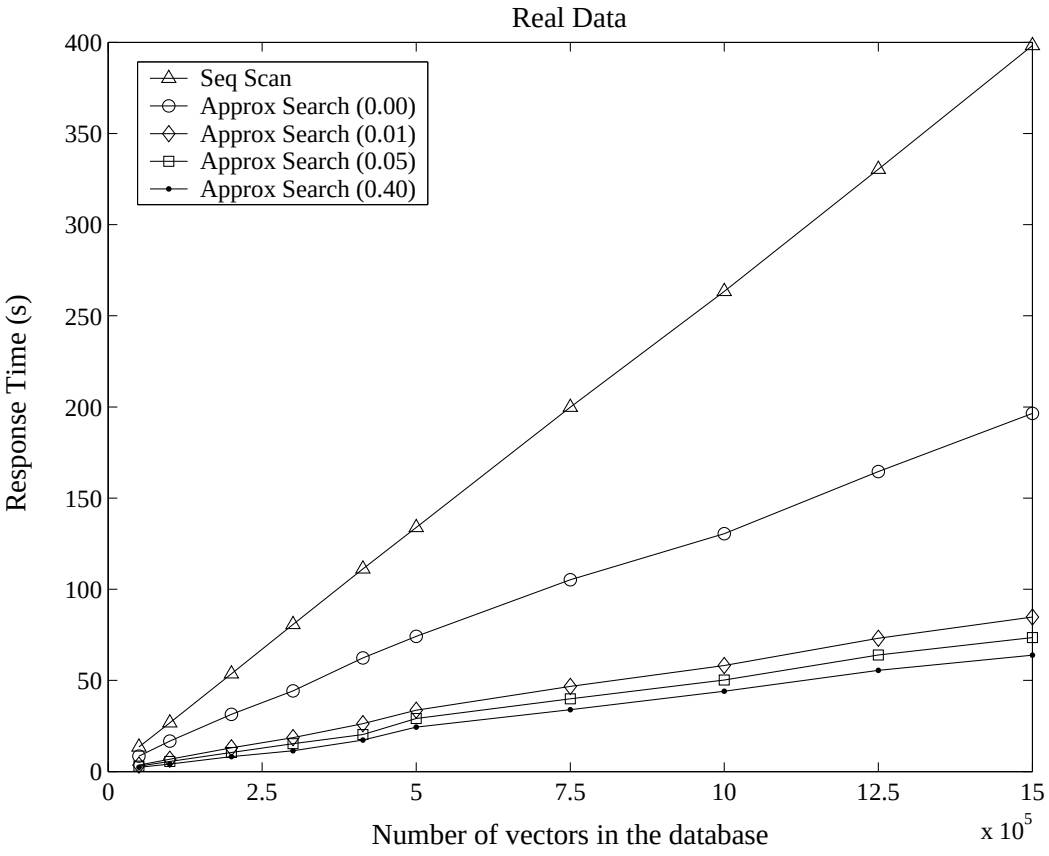

Figure 5: 24d, 200 q. vectors, 15 NNs, increasing the size of the DB.

6 Conclusion and Perspectives

This paper presents a new approach for performing efficient approximate nearest neighbor searches in high-dimensional databases. The level of precision is given at run time, and can range from lossless (e.g., exact) searches, to lower precision searches. This approach trades result quality for reduced query execution time.

Our approach assumes that most of the vectors are enclosed in clusters. Vectors not in clusters are outliers, and are treated separately. Clusters are analyzed off-line to derive, from their distribution and exact bounding hypersphere, a set of approximated bounding hyperspheres corresponding to various precision levels. Approximated radiuses are computed

on a per cluster basis, and such that the global precision of the final answer is enforced. The performance evaluation of this approach shows it is more than 5 times faster than the sequential scan when it handles more than 1.5 10^6 24-dimensions vectors, even when the probability of missing one of the true nearest-neighbors is below 0.01, and more than twice faster in the case of a lossless NN-search.

This approach currently makes some specific assumptions that must be carefully studied. Vectors are supposed to be clustered, which, in turn, gives high importance to the quality of the clusters on which our technique relies. Regardless of any technical issue, the quality of a clustering heavily depends on the nature of the data to cluster. Data following a Gaussian or a uniform distribution do not fit well in clusters. It is therefore likely that our approach fails on ill-clustered data sets. Extremely poor clustering, however, probably do not exist when considering real image descriptors computed over real data sets. We plan, however, to clarify the range of data and descriptors our approach can efficiently handle.

Our approach also assumes specific metrics (Euclidean), shape for the clusters (hyperspherical) and distributions (isotropy), that we plan to relax. The shape of clusters is directly related to the distance used to compare the vectors. It would certainly be useful to adapt the method to L_1 or L_∞ distances and thus to consider hyperrectangles or hyperdiamonds. Coping with general metric spaces is probably harder. Section 4.4 pointed out a method to cope with anisotropic vector distributions within clusters. Determining if the isotropy assumption holds for all clusters is inherently difficult as it is the case for all distribution estimations in high-dimensional spaces. If we assume by default that clusters are all anisotropic (therefore setting $P(ID_i) = 0$), then the search will prefer safety (loosing as few points as possible) over speed. Further investigations are required to make full use of the flexibility steaming from this assumption.

Focusing again on clusters raises interesting issues. Because vectors are clustered according to their similarity, it is likely that several nearest neighbors of a query points all belong to the same cluster. Therefore, the chances of missing all of them when missing only one are quite high. It is required to investigate further this issue.

Finally, evaluating the recognition capabilities of a system using approximated approaches is inherently difficult. The results depend obviously on the intrinsic power recognition of the descriptors, on the content of the DB and on the imprecision caused by the approximated scheme. It is unfair to evaluate this or that particular technique relying on approximations without also checking the potential applications.

Our approach was originally designed for the specific case of local descriptors in which the image recognition is not based on a one-to-one match between a unique descriptor stored in the DB and a single descriptor used in the query. Rather, some images stored in the DB will belong to the final answer because several descriptors of the query matched with several descriptors associated to these images. This redundancy limits the consequences of the above effects.

References

[AG01] L. Amsaleg and P. Gros. Content-based retrieval using local descriptors: Problems and issues from a database perspective. *Pattern Analysis and Applications*, 4(2/3):108–124, 2001.

[AGB01] L. Amsaleg, P. Gros, and S. Berrani. A robust technique to recognize objects in images, and the db problems it raises. In *Proc. of the 7th Workshop on Multimedia Information Systems*, 2001.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the* ACM, 45(6):891–923, November 1998.

[BFG99] K. P. Bennett, U. Fayyad, and D. Geiger. Density-based indexing for approximate nearest-neighbor queries. In *Proceedings of the 5th* ACM *International Conference on Knowledge Discovery and Data Mining, San Diego, CA USA*, pages 233–243, August 1999.

[BKK96] S. Berchtold, D. Keim, and H. Kriegel. The X-tree : An index structure for high-dimensional data. In *VLDB*, 1996.

[CP00] P. Ciaccia and M. Patella. Pac nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA*, pages 244–255, February 2000.

[FRKV94] L. Florack, B. Romeny, J. Koenderink, and M. Viergever. General intensity transformation and differential invariants. *Journal of Mathematical Imaging and Vision*, 4(2), 1994.

[GFB01] P. Gros, R. Fablet, and P. Bouthemy. New descriptors for image and video indexing. In H. Burkhardt, H.P. Kriegel, and R. Veltkamp, editors, *State-of-the-Art in Content-Based Image and Video Retrieval*, volume 22 of *Computational Imaging and Vision*, pages 213–231. Kluwer Academic Publishers, 2001.

[Hen98] A. Henrich. The LSD^h-tree: An access structure for feature vectors. In *ICDE*, 1998.

[HKM+97] J. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, 1997.

[HS95] G. R. Hjaltason and H. Samet. Ranking in spatial databases. In Egenhofer and Herring, editors, *Proceedings of the 4th International Symposium on Spatial Databases, Portland, Maine, USA*, pages 83–95, August 1995.

[LCGMW01] C. Li, E. Chang, H. Garcia-Molina, and G. Wiederhold. Clustering for approximate similarity search in high-dimensional spaces. IEEE *Transactions on Knowledge and Data Engineering*, 2001.

[RKV95] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the* ACM SIGMOD *International Conference on Management of Data, San Jose, California, USA*, pages 71–79, May 1995.

[SM97] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 19(5), 1997.

[WB00] R. Weber and K. Böhm. Trading quality for time with nearest neighbor search. In *Proceedings of the 7th Conference on Extending Database Technology, Konstanz, Germany*, pages 21–35, March 2000.

[WJ96] D. White and R. Jain. Similarity indexing with the SS-tree. In *ICDE*, 1996.

[WSB98] 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 24th International Conference on Very Large Data Bases, New York City, New York, USA*, pages 194–205, August 1998.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the* ACM SIGMOD *International Conference on Management of Data, Montreal, Canada*, pages 103–114, June 1996.

Contents

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399