CEDRIC Research Report n^{o} 1049

Back to the curse of dimensionality with local image descriptors

Nouha Bouteldja, Valérie Gouet-Brunet and Michel Scholl

CEDRIC/CNAM
292, rue Saint-Martin
F75141 Paris Cedex 03

nouha.bouteldja@cnam.fr, valerie.gouet@cnam.fr, scholl@cnam.fr

July 20, 2006

Abstract

In this report, we are interested in the fast retrieval, in a large collection of points in high-dimensional space, of points close to a query point: we want to efficiently find the set of points within a sphere of center query point p_i and radius ϵ (*a sphere query*). It has been argued that beyond a rather small dimension ($d \geq 10$) for such sphere queries as well as for other similarity queries, sequentially scanning the collection of points is faster than crossing a tree structure indexing the collection (the so-called curse of dimensionality phenomenon). The contribution of this report is to experimentally show that in the presence of redundancy in data, the curse of dimensionality is delayed to higher dimensions, rendering the use of tree-structured index still effective for a large number of applications dealing with points of moderate dimensions. We compare the performance of a single sphere query when the collection is indexed by a tree structure (an SR-tree in our experiments) to that of a sequential scan and to that of the VA-File which is an amelioration of the sequential scan. This study is applied to content-based image retrieval where images are described by local descriptors based on points of interest. Such descriptors involve a relatively small dimension (in general up to 30) and several sphere queries that are usually time consuming, justifying that the collection of points be indexed by a tree structure. The experiments were performed over 30 databases involving different synthetic and real distributions.

Keywords: CBIR, Local Descriptors, Multidimensional Indexing, Tree structure, Curse of Dimensionality, VA-File.

1 Introduction

During the last decade, similarity search has drawn considerable attention in multimedia systems, decision support and data mining. In these systems, there is the need to find a small set of objects which are similar to a given query object. Content based image retrieval is one example application where the objects to be retrieved are images, videos or parts of images. Usually, similarity is not measured on the objects themselves but on object abstractions called *features* which are points in some high-dimensional space. The number of dimensions may vary from moderate (8 to 30) to large (64 to 300) or over 900 in some applications such as astronomy. The more pertinent the object abstraction or *description*, the better the estimation of the similarity of two objects by the similarity of the features extracted from the objects. The similarity of two features (points) is a function of their distance in the high-dimensional space. Efficient similarity search is supported by a multidimensional index structure.

In this report we are interested in the fast retrieval, in a large collection of points in a high-dimensional space, of points close to a query point. We want to efficiently find the set of points within a sphere of center query point p_i and radius ϵ (ϵ-sphere). It has been argued [35] that beyond a rather small dimension ($d \geq 10$) for such sphere queries as well as for k-NN queries, sequentially scanning the collection of points is faster than crossing the tree-structured multidimensional index: we briefly survey the well-known *curse of dimensionality phenomenon* [25, 6, 34] in section 2.3. The more uniform the point distribution in space, the smaller such a threshold dimension. In [34] Weber and al. formally shown that the curse of dimensionality is reached for index structures with a dimension larger than 10 [35] but under the assumptions of uniformity and independence of the dataset. They proposed the VA-File as an improvement of sequential scan.

Here, we want to experimentally assess whether the curse of dimensionality is reached with various points distributions when the number of dimensions is moderate. Considerable attention has been devoted to data structures that delay the curse of dimensionality but surprisingly no study experimentally evaluates until which dimension a tree structure is still appropriate. We compare the performance of a single sphere query when the collection is indexed by a tree structure (an SR-tree in our experiments) to that of a sequential scan and to that of a VA-File. We performed an extensive performance evaluation over 30 synthetic and real datasets. As an example of real data, the points were obtained by the description of the visual content of images.

Outline of the report The following section gives a survey on related multidimensional access methods with emphasis on the SR-Tree and the VA-File structures and revisits the curse of dimensionality phenomenon. Section 3 is dedicated to content-based image retrieval, and more precisely to local image descriptors, which will constitute the real datasets of the evaluation. Section 4 describes our experiments and provides some evaluation results: we assess the use of an SR-tree for single sphere queries in applications where the objects are described by points in a moderately high-dimensional space.

2 Related work on multidimensional access methods

Several multidimensional access methods (MAM) have been proposed in the past decades. They mostly differ in the manner data are partitioned in space, see for example [7] and [23] for a more recent classification of these methods. A large number of dynamic MAM are variants of the R-Tree [15] where the collection of points is organized in a hierarchy of hyper rectangles. These variants differ in the strategy chosen for splitting data; a survey on R-Trees can be found in [21]. We propose the use of the SR-tree [18] structure, which is a combination of the R*-tree [1] (one of the most efficient R-Trees) and the SS-Tree [36] (a hierarchy method based on bounding hyper spheres). The SR-tree structure is described in subsection 2.1 where an algorithm for sphere queries is also introduced. If the dimension of the space is high, MAM's performance suffers from the well-known curse of dimensionality phenomenon [25]; the latter is briefly surveyed in subsection 2.3. Arguing that if the space dimension is above a given dimension, sequential scan outperforms hierarchical index traversal, Weber and al. [25] proposed the VA-File, as an amelioration of sequential scan. We revisit the VA-file structure in subsection 2.2 and also give an algorithm for the sphere query. Many improvements of the VA-File have recently been proposed: among them, it is important to note the VA-file parallelization [33], the KVA-File [16] that extends VA-File to Kernel-based retrieval methods and the VA$^+$-File [11] which performs a transformation on the data space to ameliorate the VA-File with non uniform distributions. Other techniques combine the tree structure with a VA-file like approximation, for example the IQ-Tree [3] and the A-Tree [27]. The former uses approximations in the leaves of an R*-tree-like structure while the latter uses approximations all over the tree structure; it is shown that the A-tree outperforms both the VA-File and the SR-tree. In [13], a technique for taking advantage of both the A-Tree and the IQ-Tree is proposed to improve even more search in high-dimensional data spaces.

The rest of this section is devoted to the description of the SR-tree, the VA-File and to a discussion on curse of the dimensionality phenomenon.

2.1 SR-tree

We have chosen to use the SR-tree [18] as an index structure for high-dimensional sphere queries, because from one hand, its code[1] was available on the line at the time when the experiment started, and from another its performance is by far better than that of the regular R*tree [1] for high dimensions. We chose to compare its performance to a simple approximate based sequential scan, namely the VA-file. One might argue that it would be fairer to compare the performance of a tree structure such as the A-tree or the IQ-tree to that of an improved VA-file. However one should remember that our target is not to decide which data structure is the best, such decision would depend on many parameters such as the data distribution and the dimension. In contrast, our objective is to verify whether the curse of dimensionality is reached for moderate dimension and some realistic data distributions. It is our belief that similar results and conclusions would hold if we were to compare say the performance of an X-tree [5] or that of the Pyramid-tree [4] or that of an A-tree with the performance of an improved VA-file. In the following, we recall the principle of SR-tree.

SR-tree principle

The structure of the SR-tree is a mixture of that of the R*tree [1] and that of the SS-tree [36]. The SR-tree is illustrated in Figure 1 for a 2-dimensional space. It is a hierarchy of regions where regions in a subtree with root r are included in the region associated with r. Each region is specified by the intersection of the minimum bounding sphere and the minimum bounding rectangle of the included regions (internal nodes of the tree) or points (leaves of the tree). A leaf is a set of n vectors (points), where the maximum value of n depends on the page capacity. An internal node N has the following structure: $[C_1, ..., C_n]$ where C_i corresponds to a child of the node and is the tuple $[S, R, w, c_p]$ where S is its bounding sphere, R the bounding rectangle and w the total number of vectors in the subtree whose root is the child pointed by c_p.

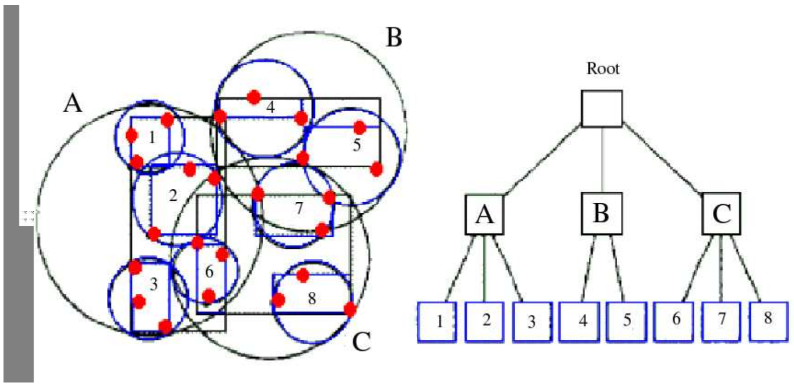


Figure 1: Example of structuring with SR-tree.

For a detailed construction of the tree, see [18]. By employing spheres as regions, the SS-tree [36] was shown to significantly outperform the R*tree in the case of similarity search. However the sphere volume is large, which leads to a large overlap between nodes when the dimension increases. By taking the intersection with the bounding rectangle we keep the advantage of the sphere (small diameter wrt

[1] $http://research.nii.ac.jp/\ katayama/homepage$

to rectangles) and have a much smaller volume for the region (the smaller the region volume, the less overlap with twin regions).

SR-tree with sphere queries

Because initially proposed for k-NN search, we developed code for sphere queries on top of the available SR-tree code. The algorithm for sphere query, denoted $S(T, q)$, is described in Algorithm 1. T is the tree root page and q is a sphere query (a sphere with center p and radius ϵ). It is similar to the straightforward depth-first traversal of an R*tree.

Algorithm 1: S(T,q)

```
Input  : T a tree root, q a sphere query (sphere with center p and radius $\epsilon$)
Output: V a set of vectors $< V_1, V_2, ..., V_m >$
// Computes the set V of vectors of tree with rootpage T inside the sphere q
V:= empty;
readpage T;
if T is not a leaf then
    for each child C in T do
        if C.R intersects q and C.S intersects q then
            // C.R (C.S) minimum bounding rectangle (sphere) of C
            V+= {S(C, q)};
    end for;
else
    for each point t in T do
        if t in q then then V+={t};
    end for;
end if;
return V;
```

2.2 VA-File

As a second multidimensional access method, we choose the VA-File because it ameliorates sequential scan even for very large dimensions. Its code was also available on the line. The VA-File (Vector Approximation File) is based on a concise representation of a vector by an approximation [32]. The approximations of each vector of the dataset are concatenated to generate a vector approximation file (VA-file) which is compact enough to hold in central memory, whereas the data vectors are saved in second memory in an other file: the "Basic File" (or "Vector File"). When searching points similar to a query point, the VA-file is first sequentially scanned entirely to exclude a part of vectors (Filtering step). A superset of the solution is then chosen only on approximations of the vectors. After this step only a small subset of pages of the vector file are loaded from second memory in order to check for similarity on the vector themselves. The VA-File structure is illustrated in Figure 2 with two dimensions. The whole structure is composed of two files. The high dimensional space is divided into cells such that each cell contains at most one point with very high probability. The larger the dimension, the smaller the number of non empty cells. Each point of the dataset has two representations: the regular precise one and its approximation by the identity of the cell it belongs to. For example, the approximation of point $(0.1, 0.9)$ is 0011.

We have adapted the VA-File code for sphere queries. This version of the algorithm, denoted $VA(F, q)$, assumes that there is one vector per case, which is the most probable. It is described in Algorithm 2. F is a VA-File composed of two files, the vector and the approximation ones, and q is a sphere query (a sphere with center p and radius ϵ). We begin by sequentially scanning the approximation file and applying two filtering rules:

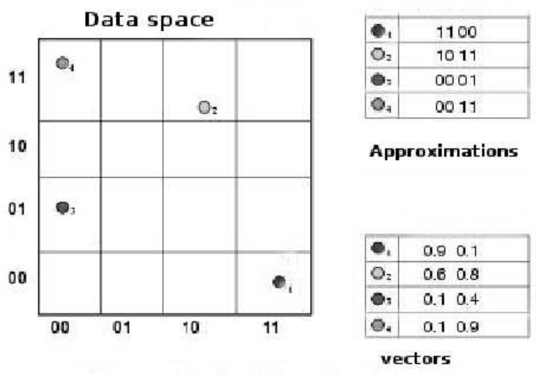


Figure 2: Example of structuring with VA File.

- All the vectors whose approximation case is totally included in the query sphere are added to the solution without computing their distances to the query,
- The vectors whose approximation does not intersect the ϵ sphere are discarded.

For the remaining approximations (which intersect the query but not fully included into the sphere) the corresponding vectors have to be loaded from secondary memory in order to check for inclusion of the vector in the sphere.

Algorithm 2: VA(F,q)

```
Input  : F a VA-File contains an approximation file AF and a vector file VF, q a sphere query
         (sphere with center p and radius $\epsilon$)
Output: V a set of vectors
// Computes the set V of vectors of VA-File F inside the sphere q
V:= empty;
for each approximation C in AF do
    // Filtering
    if C in q then then
        read the vector v from VF corresponding to C;
        V+={v};
    else
        // Filtering
        if q intersects C then then
            read the vector v from VF corresponding to C;
            if v in q then then V+={v};
    end if;
end for;
return V;
```

2.3 Curse of dimensionality

The concept of "curse of dimensionality" was first coined by Richard Bellman [25]. He employed it to describe the problem caused by the exponential increase of the volume with the augmentation of the space dimension when addressing the problem of optimizing functions with several variables. Later, the term was used to indicate, more generally, nonintuitive phenomena observed when the dimension of data increases. Among these phenomena, let us quote the most important ones (for more details, the reader can see [6, 34, 7, 30]):

- The *"exponential increase of the space volume"* phenomenon, which has many effects: the number of partitions generated by the MAM based on space partitioning grows dramatically which has for a consequence a drastic performance decrease of these structures. Related to this, space becomes "sparsely populated" i.e. the probability to find a point in a sphere or a rectangle, with a width less than the space one, gets very small when the dimension increases.
- The *"measures concentration"* phenomenon, which is related to the distances between vectors: in high-dimensions, distances between objects become almost identical. Under these conditions, the noise coming from data acquisition disturbs too much the usual metrics. As another consequence of the "measures concentration", k-NN search is not anymore meaningful since the distance of a query point to the nearest neighbor is closer to the distance to the farthest one [6].
- The *"concentration of the volume on the surface"* phenomenon: the volume of an hyper-rectangle placed in the center of the space and with an edge equal to 0.99 % of the space edge, is very small compared to the space volume in high dimensions; for example it represents $\simeq 13.39\%$ in dimension 200.

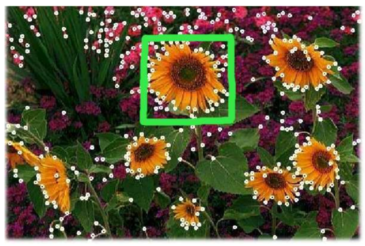
As the dimension increases, the aforementioned effects increase enough to have a negative impact on performance. One may then wonder from which dimension this curse of dimensionality phenomenon is reached. Several responses to this question were given in the literature:

- Verleysen and al. [31] for neuronal networks, Weber and al. [34] for MAM methods, show that an exhaustive scan is faster than a tree traversal from a dimension of about 10.
- Beyer and al [6] demonstrate that k-NN search becomes meaningless from dimension 20 (under some assumptions such as data uniform distribution in space), because of the "measures concentration" phenomenon.
- Verleysen et al. [29] link the curse of dimensionality problem to the number of points composing the database. They assume that a space is high dimensional when the size of its population does not grow exponentially with its dimension.
- More recently, Verleysen and François [30] observed that redundancy in data coordinates is necessary for analyzing data in high-dimensional space. They showed that in the presence of a small database in a high dimensional space various methods (like linear regression) hardly analyze such data; redundancy could remediate to the cited problem.

In the present study, the curse of dimensionality problem is said to be reached if a tree traversal is not worth anymore, i.e. if its performance is lower than that of a sequential scan and show that the distribution and the size of the dataset has a significant impact on this phenomenon. It has been argued that beyond a rather small dimension ($d \geq 10$) sequentially scanning the collection of points is faster than traversing the tree-structured multidimensional index [25, 6, 34, 35]. The more uniform the point distribution in space, the smaller such a threshold dimension. In [34], Weber and al. formally show that the curse of dimensionality is reached for index structures with a dimension larger than 10 but under the assumptions of uniformity and independence of the dimensions of the vectors composing the dataset. To the best of our knowledge, no formal or experimental study has yet evaluated or estimated until which dimension a tree structure is still appropriate, when varying the data distribution.

3 Image retrieval with local descriptors

Descriptors dedicated to Content-Based Image Retrieval (CBIR) mainly depend on the applications and datasets of images considered. When considering *global* or *approximate* image retrieval, most of the descriptors are based on the global color, texture and shape of the image. When dealing with sub-image retrieval, *local* descriptors are a very popular approach in the CBIR community. In particular, they have smart properties that make them relevant for object recognition[2] or more generally sub-image retrieval in datasets of heterogeneous images, as illustrated in Figure 3.


(a) Query support

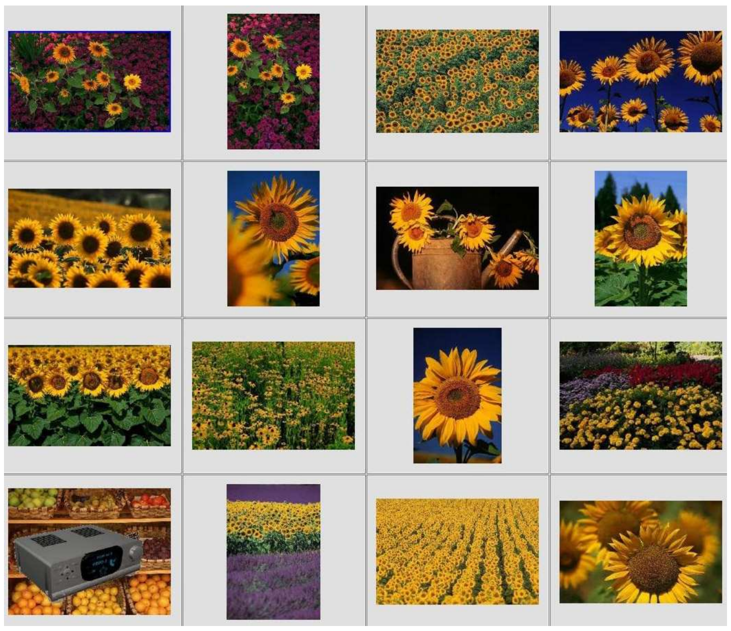

(b) Responses

Figure 3: *Example of object retrieval using local descriptors. (a) is the query support presented with extracted points of interest (in white). The query is a sunflower that has been manually determined by the user (bounding rectangle). (b) shows the 16 first retrieved images from a data set of generic images, presented by decreasing order of similarity. These results were obtained with the CBIR system IKONA, by courtesy of INRIA (http://www-rocq.inria.fr/imedia/ikona.html).*

In this paper, we focus on such descriptors, because similarity search with local descriptors implies multiple queries, i.e. several points in the query, making on-line retrieval more time consuming than with global descriptors and then justifying our study.

Local descriptors computation

Local descriptors are usually based on *points of interest* extraction and local characterization. Such an approach allows to gain robustness against occlusions and cluttering since only a local description of the patch of interest is involved. It is said to be *salient features-based*, because the information extracted from the image is condensed into limited but salient sites.

Image retrieval based on local descriptors follows a classical computation flow: first a robust salient feature detector must be designed. Such a detector determines the location of salient points in the image, plus the support regions around them where the local descriptor is to be computed. Second, a rich and compact descriptor is computed for each salient point, by analyzing the image information within the support region exhibited. Finally, a similarity measure must be found to compare two salient point signatures.

Many techniques of local descriptors have been proposed in the literature of Computer Vision and CBIR, see for example [28, 20, 14, 2, 10] or the performance evaluation in [22]. Approaches depend on the considered application and more precisely on the image transformations authorized.

[2]See for example the report of the European PASCAL Challenge on Visual Object Classes recognition [10], where most of the approaches proposed are based on local descriptors.

Dimensionality of local descriptors feature spaces

When considering similarity search, such local approaches differ from global ones by the *dimensionality* of the feature space involved: the content of the image is represented by *several vectors* in a *relatively low-dimensional (moderate) feature space.* In comparison, global descriptors involve only *one vector* in a *high-dimensional space.* Characteristics of feature spaces for some global and local popular descriptors are given in table 1.

Descriptor	Category	Size/image	Dimension
Color histogram	global	1	>200
Gabor filters	global	1	~48
Regions of interest	local	4-20	~20
Points of interest	local	100-500	8-30, 128

Table 1: *Examples of popular content-based image descriptors. The sizes (numbers of vectors per image) and the dimensions given are representative of the descriptors considered.*

As shown in the table, a majority of the encountered local descriptors involve a feature space having between 8 and 30 dimensions or 128 dimensions for the popular SIFT descriptor [20] or the GLOH one [22]. Because of their dimension, these last descriptors suffer from curse of dimensionality problem, making sequential scan more efficient than using an index structure (as explained in section 2.3). The encountered implementations of such descriptors apply a reduction of their size to remain efficient through an index structure during similarity search. Reduction is usually performed with a Principal Component Analysis: in [19], the best trade-off (in terms of precision/recall and time retrieval) is obtained for dimensions of the reduced SIFT between 20 and 36, as also experimented in [24]. In [9], only the first 5 best components of SIFT are kept during search through a kD-tree.

According to these observations, we will consider dimensions from 8 up to 30 in our experiments. This range of values is representative of the implementations aforementioned, that aim at practicable on-line similarity search on large datasets.

Similarity search with local descriptors

Image retrieval with local descriptors implies a specific algorithm for similarity search that proceeds in several steps:

1. For each query point $p_{i,i\in\{1,m\}}$, find in the feature space a set of neighbors points called $N(p_i)$;
2. For each image I_j of the database, compute a vote v_j, that is a function of the number of neighbors retrieved and belonging to I_j and of their associated distances. A similar image is associated with a vote large enough that expresses that it contains a lot of points close to most of the query points;
3. Finally, the images are usually sorted by decreasing order of their vote, as shown in Figure 3.

Finding the nearest neighbors $N(p_i)$ of step #1 can be done according to two strategies [7]:

- Either a k-NN search ("Retrieve the k images most similar"). Here, the advantage is that k can be easy to determine but the drawback is that some of the retrieved images may not be similar to the query image;
- Or an ϵ-sphere query, where ϵ can be estimated ("Retrieve all the similar images") of fixed by the user ("Retrieve all the similar images up to a distance ϵ"). Generally the difficulty lies in determining the threshold ϵ.

With local descriptors, the strategy usually follows an ϵ-sphere query. The k-NN strategy is not well suited because k may be very different for each query point p_i and not easy to determine intuitively as for global descriptors. If ϵ is known or estimated, the set $N(p_i)$ is defined by:

$$N(p_i) = \{n_k \in F_d \ / \ dist(p_i, n_k) < \epsilon\}$$

where $dist(.,.)$ is the similarity measure associated with the feature space F_d of dimension d. Traditionally for local descriptors, the similarity measure associated is the distance L_2 or more generally the Mahalanobis distance δ_2. In the rest of the paper, q_i will denote the sphere query of center p_i and radius ϵ, and $d_{p_i p_j}$ the measure $dist(p_i, p_j)$ between two points p_i and p_j. It is worth noting that for our experiments, we have an exact idea of the values of ϵ which yield good results for the databases extracted from video images that will be used in the experiments.

4 Experimentations

We performed an extensive experimental evaluation of the performance of SR-trees for sphere queries, using synthetic and real data based on local descriptors. The distributions of datasets and queries employed are described in section 4.1. All the experiments were performed on an Intel Pentium (3.2 G Hz) based workstation running Linux with 1G bytes of main memory. The algorithms and data structures were implemented in C++. We used the SR-tree code version 2.0 beta 5 provided by the authors. Each SR-tree is built dynamically and we stress that its structure and the data are not loaded in main memory: during retrieval, each node is read from the disk, when necessary. For the VA-File we utilized code provided by the authors with default parameters.

4.1 Tested datasets

The experiments were done over 30 different datasets, split into two main categories:

- Datasets composed of roughly 218000 points.
- Larger datasets (referred in the following by "big" datasets) which are composed of 1 Million of points each.

For each category, we generated real and synthetic data with three dimensions that are representative of local descriptors for efficient similarity search, as explained in section 3. The main parameters and names of the 30 datasets obtained are synthesized in Table 2 and Figure 4 shows some of the distributions used, reduced for display to 3 dimensions with a PCA (Principal Component Analysis).

	Distributions				
Dim.	Real	Uniform	Clustered		
			$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.2$
8	D^8_R	D^8_U	D^8_{C1}	D^8_{C2}	D^8_{C3}
17	D^{17}_R	D^{17}_U	D^{17}_{C1}	D^{17}_{C2}	D^{17}_{C3}
29	D^{29}_R	D^{29}_U	D^{29}_{C1}	D^{29}_{C2}	D^{29}_{C3}

(a) Small Distributions (218 000 vectors).

	Distributions				
Dim.	Real	Uniform	Clustered		
			$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.2$
9	BD^9_R	BD^9_U	BD^9_{C1}	BD^9_{C2}	BD^9_{C3}
18	BD^{18}_R	BD^{18}_U	BD^{18}_{C1}	BD^{18}_{C2}	BD^{18}_{C3}
30	BD^{30}_R	BD^{30}_U	BD^{30}_{C1}	BD^{30}_{C2}	BD^{30}_{C3}

(b) Big Distributions (1M of vectors).

Table 2: *Synthetic and real datasets used for the evaluation.*

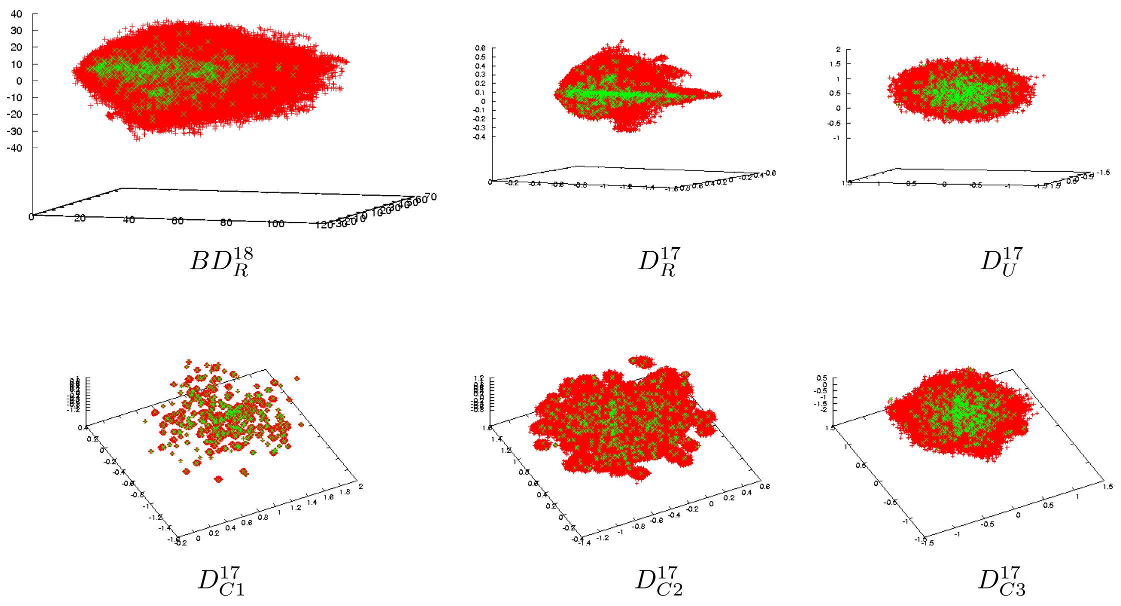


Figure 4: *Visualization of some of the distributions evaluated, reduced for display to 3 dimensions with a PCA.*

4.1.1 Small datasets

The first category (small datasets) is composed of:

- Real datasets (D_R) extracted from the well-known database of images COIL-100 [8]. This database contains 7200 color images representing 100 objects displayed under 72 viewpoints. Each point is extracted with the Harris detector and is characterized with three different feature vectors involving several orders of Hilbert color invariants [14], providing feature spaces of dimensions 8, 17 and 29;
- Uniform datasets (D_U): synthetic point coordinates are uniformly generated in the interval $[0, 1]$. Three datasets were computed for the same dimensions;
- Synthetic clustered datasets obtained for the same dimensions with different shapes. Each one has 312 clusters of 700 points each, with a randomly generated center in $[0, 1]^d$. Points in a cluster follow a Gaussian distribution with respective values of $\sigma = 0.01$ (D_{C1} sets), $\sigma = 0.05$ (D_{C2} sets) and $\sigma = 0.2$ (D_{C3} sets).

4.1.2 Big datasets

Similarly, the big datasets correspond to:

- Real datasets (BD_R) extracted from color video images of road traffic video-surveillance. Each extracted Harris point is characterized with three feature vectors based on the local set of image derivatives at different orders [17], providing three feature spaces of dimensions 9, 18 and 30. For these datasets, we have a precise idea of the optimal values of ϵ that yield good retrieval results[3], see Table 3;
- Uniform datasets (BD_U) corresponding to three synthetic uniform datasets with the same three dimensions;
- Clustered datasets obtained for the same three dimensions with different shapes. Each one has 2000 clusters of 500 points each, with a randomly generated center in $[0, 1]^d$. Points in a cluster follow a Gaussian distribution with respective values of $\sigma = 0.01$ (BD_{C1} sets), $\sigma = 0.05$ (BD_{C2}sets) and $\sigma = 0.2$ (BD_{C3} sets).

	dimension		
	9	18	30
ϵ	4.11	5.37	6.62

Table 3: *Dataset BD_R: relevant values of ϵ for each dimension.*

The similarity measure associated with all the feature vectors generated is the L_2 distance. The two kinds of local descriptors computed for D_R and BD_R are usually associated with the Mahalanobis distance, but we normalized them (change of basis by decomposition of the covariance matrix) to use the L_2 distance. To have a more accurate idea about the distributions of real datasets D_R and BD_R, we plotted in Figure 5 the histograms of distances (normalized) between points for each dataset. We observe the distributions are very different: in particular, BD_R has much more variability.

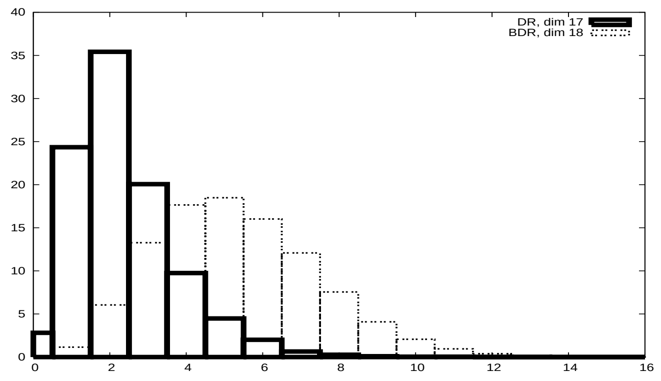


Figure 5: *Histograms of distances for the real datasets D_R^{17} and BD_R^{18}.*

4.1.3 Queries

All the experiments were done on a sample of 500 points randomly chosen among each dataset. For the real collections, the 500 points correspond to different images. Each performance measure that will be presented corresponds to an average over this sample.

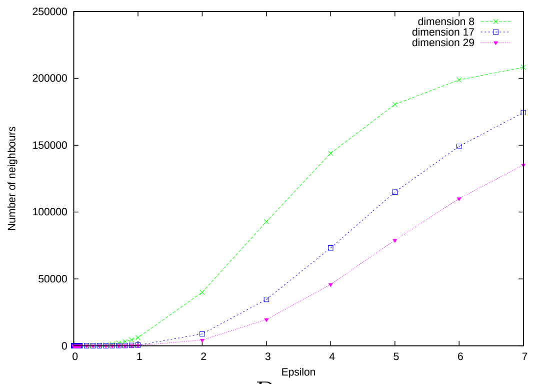
For each category of our tested datasets we will first display experiments made on the real distributions, the uniform ones and finally on the second clustered datasets (with $\sigma = 0.05$). Because of space limitations, we do not display the results for other clustered distributions but briefly discuss differences with the displayed distribution if any. Sometimes, we also present only one performance measure associated with a relevant value of ϵ: for BD_R, ϵ is taken in Table 3. For other datasets, the appropriate ϵ corresponds to a value that brings at least c neighbors (for the clustered datasets, c is the number of points per cluster, then $c = 700$ for D_{Ci} and $c = 500$ for BD_{Ci}, whereas it is about 80 for D_R since for each object there are 72 similar objects in the database). The three following sections 4.2, 4.3 and 4.4 synthesize the results obtained.

4.2 Number of neighbors according different distributions and dimensions

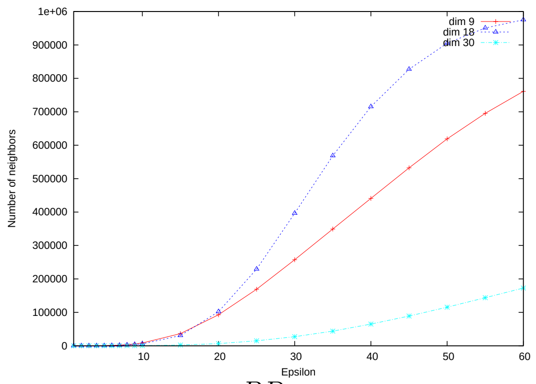
In Figures 6 and 7, we display the average number of points found inside the query sphere versus the sphere radius ϵ.

The behavior of the curves is not the same for the real datasets as for the synthetic distributions clustered and uniform whose curves look very similar (see Figure 6). As expected for the uniform distribution (this is the curse of dimensionality phenomenon [34]), all points are much further away from each other: the number of neighbors for a relatively large value of ϵ (0.4) is still 1 for D_U^{17} and BD_U^{18}. For dimension 29, one finds one neighbor at $\epsilon = 1$. In contrast, for clustered synthetic data, e.g. for

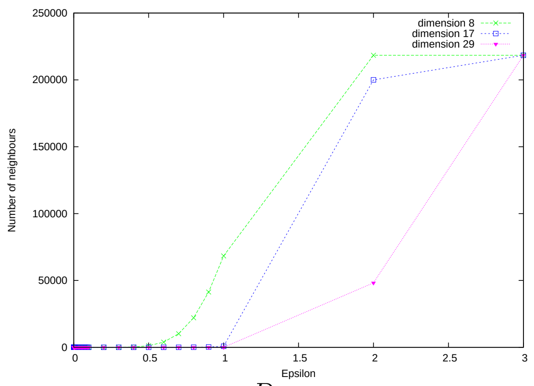
[3]We deduced them from ROC (Receiver Operating Characteristic) curves of parameter ϵ computed during experiments conducted in parallel for object recognition purposes, over these videos and with these descriptors.



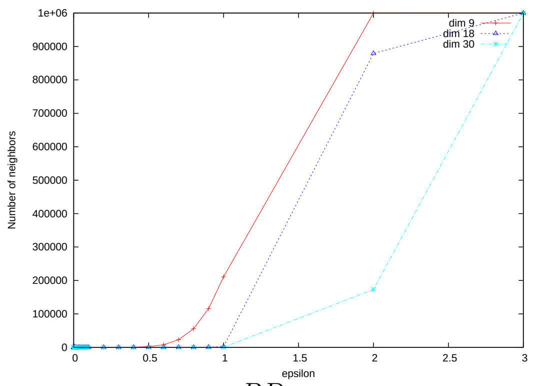
D_R



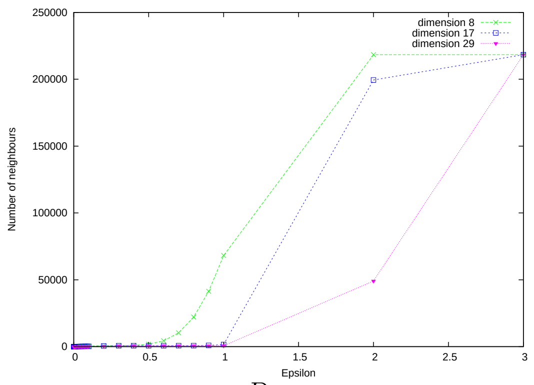
BD_R



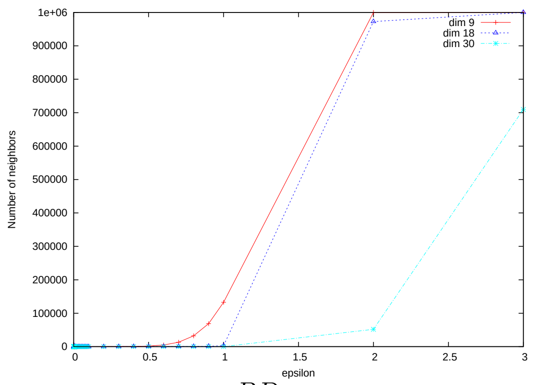
D_U



BD_U



D_{C2}



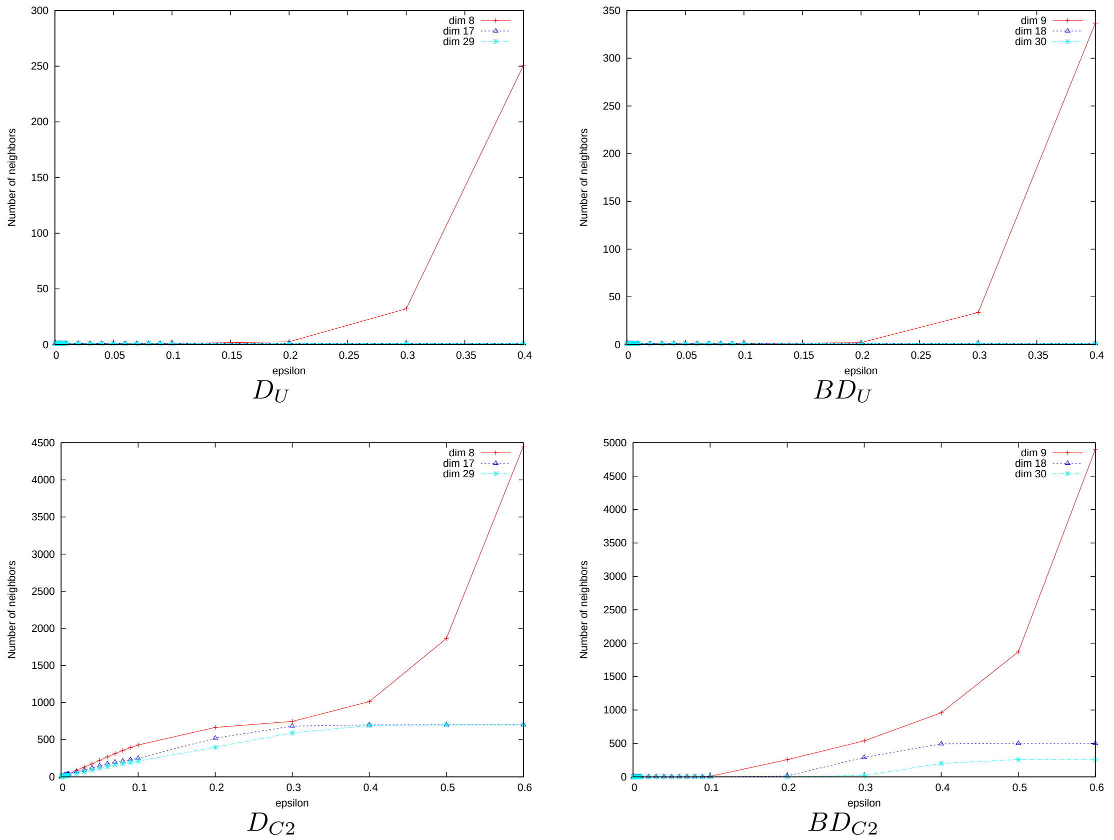
BD_{C2}

(a) Small datasets (b) Big datasets

Figure 6: *Number of neighbors versus ϵ (a zoom is given in Figure 7 for the synthetic datasets).*

the clustered distribution D_{C2} (Figure 7), for large dimensions, with a value of $\epsilon = 0.3$ all points of the cluster corresponding to the query point are within the sphere.

The curves of the big and the small real distributions are also different: with the small real dataset D_R (Figure 6), for dimensions 8, 17 and 29, respectively 608, 20 and 8 neighbors are found when $\epsilon = 0.5$, while with $\epsilon = 1$, one respectively gets 6062, 465 and 166 points. With a point query chosen in the dataset, a "pertinent" value of ϵ is one such that there are approximately 100 neighbor objects within the query sphere (recall that the dataset includes objects under 72 different viewpoints). Therefore a pertinent value of ϵ is less than 0.5 for dimension 8 and less than 1 for dimension 29. In contrast with the real dataset BD_R, a pertinent value of ϵ (see Table 3) is ten times the one of the small dataset.



D_U BD_U

D_{C2} BD_{C2}

(a) Small datasets (b) Big datasets

Figure 7: *A zoom of the number of neighbors for the synthetic datasets.*

Has the curse of dimensionality been reached?

In summary, as expected, the number of neighbors within a sphere significantly depends on the data distribution, and of course is independent from the search method (SR-tree, Sequential or VA-File traversal). One can say that, except for the uniform distribution, even for dimension 29, the curse of dimensionality has *not* been reached. Indeed, the query points are selected from the databases and:

- For the clustered datasets, they belong to one of the clusters composing the datasets. As aforementioned, we are sure that we will find all the points composing the cluster to which this query belongs, at a distance at most equal to the cluster diameter. Note that, when ϵ increases we get a behavior similar to that of the uniform. Indeed the cluster centers are uniformally distributed.
- With the uniform dataset, the larger the database size, the larger the value of ϵ in order to get a single neighbor and by enlarging the value of ϵ we get nearly all the database points (the "measures concentration" phenomenon holds, i.e. the distance to the nearest neighbor is almost the same as that to the farthest one).
- As expected, for the real dataset, even for a dimension of 29, although not as clustered as for the synthetic dataset, data is not sparsely distributed as for the uniform dataset.

Is a tree-structured index still appropriate for sphere queries when the dimension is larger than 10? For which data distributions? Sections 4.3 and 4.4 attempt to answer these questions by respectively comparing the sphere query time performance of an SR-tree to that of a sequential scan and to that of a VA-file.

4.3 Speed-up SR-tree/Sequential

It is important to stress that in the experiments conducted in this paper, we measure only CPU time, since we are CPU bound, i.e. data holds in central memory. When appropriate we compare the number of block accessed as an indication of a possible behavior for larger data sizes. Although we could have investigated the impact of data size more exhaustively for synthetic data, we have left this as a future work since very large real life data were not available at the time of the experiments. For space limitations we do not show here the results with the uniform distributions: as expected, there is a very small gain with a dimension $d = 17$ and no gain with dimension 29 with a uniform distribution, while there is a gain for $d < 10$. In the following the CPU gain or the speed up designates the ratio Sequential scan CPU Time over the SR-tree one.

Table 4 summarizes the results for real and clustered databases. Clearly the curse of dimensionality has not been reached for dimensions higher than 10. The clustered small dataset exhibits a gain greater than 20! For the large real dataset the gain is almost 9. It is important to note that the larger the data size, the larger the gain. The poor performance of the large clustered dataset is probably due to the poor adequation of the SR-tree parameters for this distribution.

Distribution	dimension (d)		
	$d < 10$ (8 or 9)	$d < 20$ (17 or 18)	$d \leq 30$ (29 or 30)
D_R	10.48	4.53	3.45
D_{C2}	17.64	24.40	37.41
BD_R	15.10	8.92	69.25
BD_{C2}	15.95	4.11	1.50

Table 4: *CPU time speed-up for SR-tree vs Sequential scan. The gain is given for pertinent values of* ϵ.

Figure 8 further illustrates the CPU time gain when using an SR-tree wrt a sequential scan. For both the small datasets (a) and the big datasets (b), we plot the ratio of the sequential scan CPU time over the CPU time obtained by the SR-tree traversal versus the number of neighbors (except for BD_R where the results are displayed up to the relevant values of ϵ in Table 3).

From the experiments, the following facts are noteworthy:

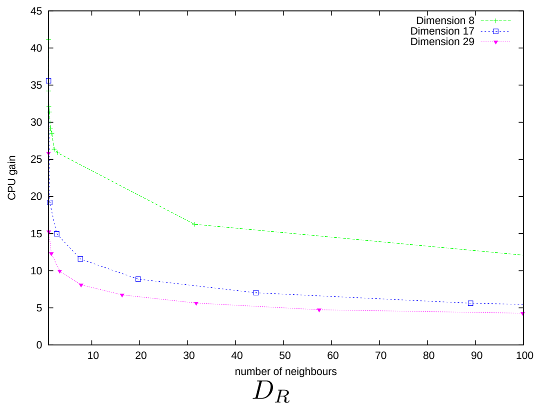
1. The extremely high gain for dimension 30 with the large real dataset is due to the clustered nature of the video images taken for this large dataset.

2. With BD_{C1}, the SR-tree is from far better than the sequential scan; for 500 neighbors (number of vectors per cluster) the gain is over 80 even for the dimension 30.

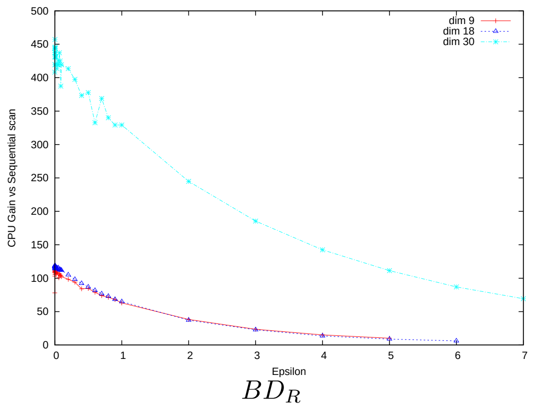
3. Changing the value of σ has an important impact on the gain: for BD_{C2} ($\sigma = 0.05$), we have also a gain but not as important as for the first clustered distribution ($\sigma = 0.01$). It varies from 16 for dimension 9 to down 3 for dimension 30.

Number of nodes acceded

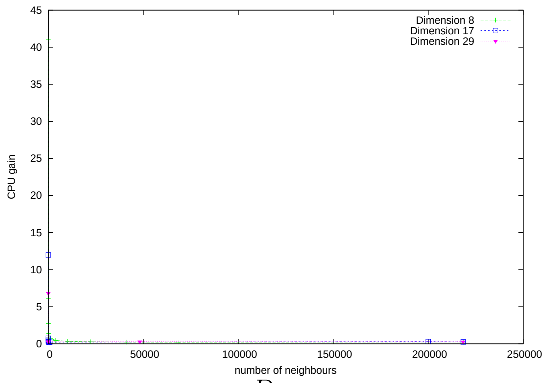
Figure 9 (with a zoom for the small uniform dataset in Figure 10(b)) plots the ratio between the number of nodes accessed during tree traversal versus the total number of nodes of the tree versus the number of neighbors for both small and large datasets. The former study on CPU time gain is confirmed by this figure: the percentage of nodes accessed is small (≤ 20) for the distributions which gave good results. As aforementioned, the atypical good performance of the large dataset for dimension 30 is related to the good clustering of the video images that lead to a cluster size pretty close to the page size.



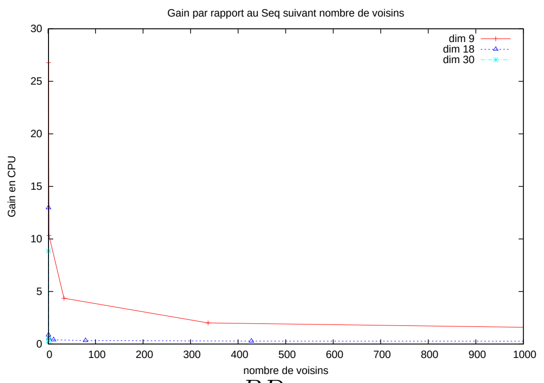
D_R



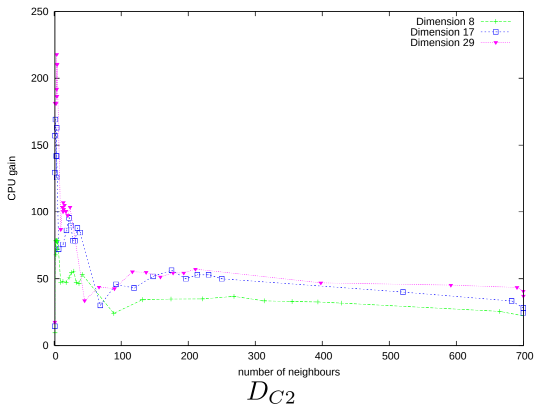
BD_R



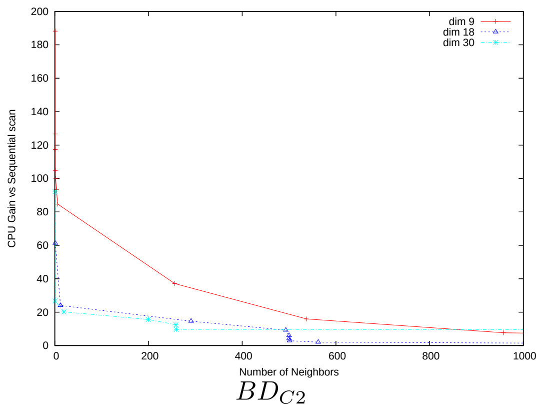
D_U



BD_U



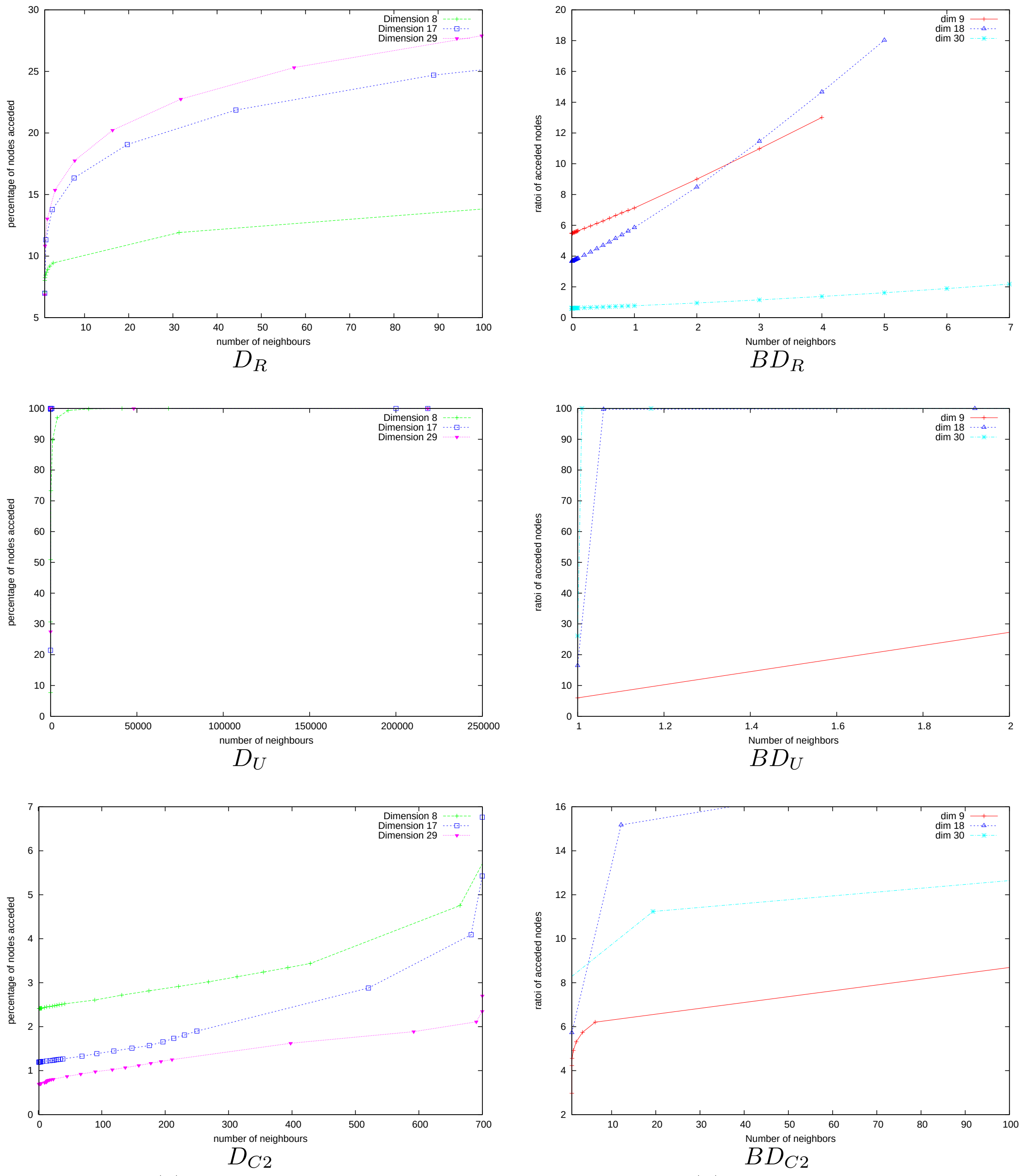
D_{C2}



BD_{C2}

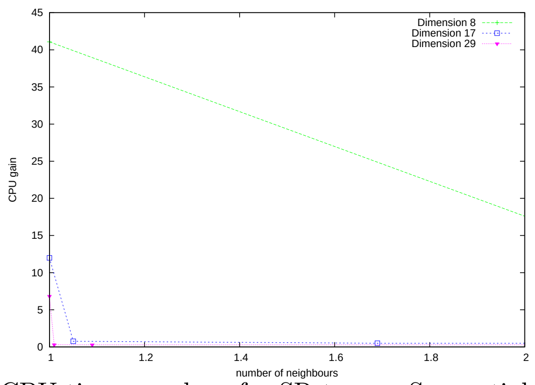
(a) Small datasets (b) Big datasets

Figure 8: *CPU time speed-up for SR-tree vs Sequential scan (a zoom is given in Figure 10.(a) for the uniform datasets).*

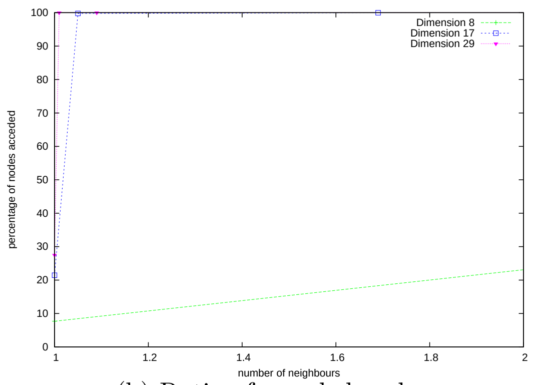


(a) Small datasets (b) Big datasets

Figure 9: *Ratio of acceded nodes (a zoom is given in Figure 10.(b) for the uniform datasets).*



(a) CPU time speed-up for SR-tree vs Sequential scan

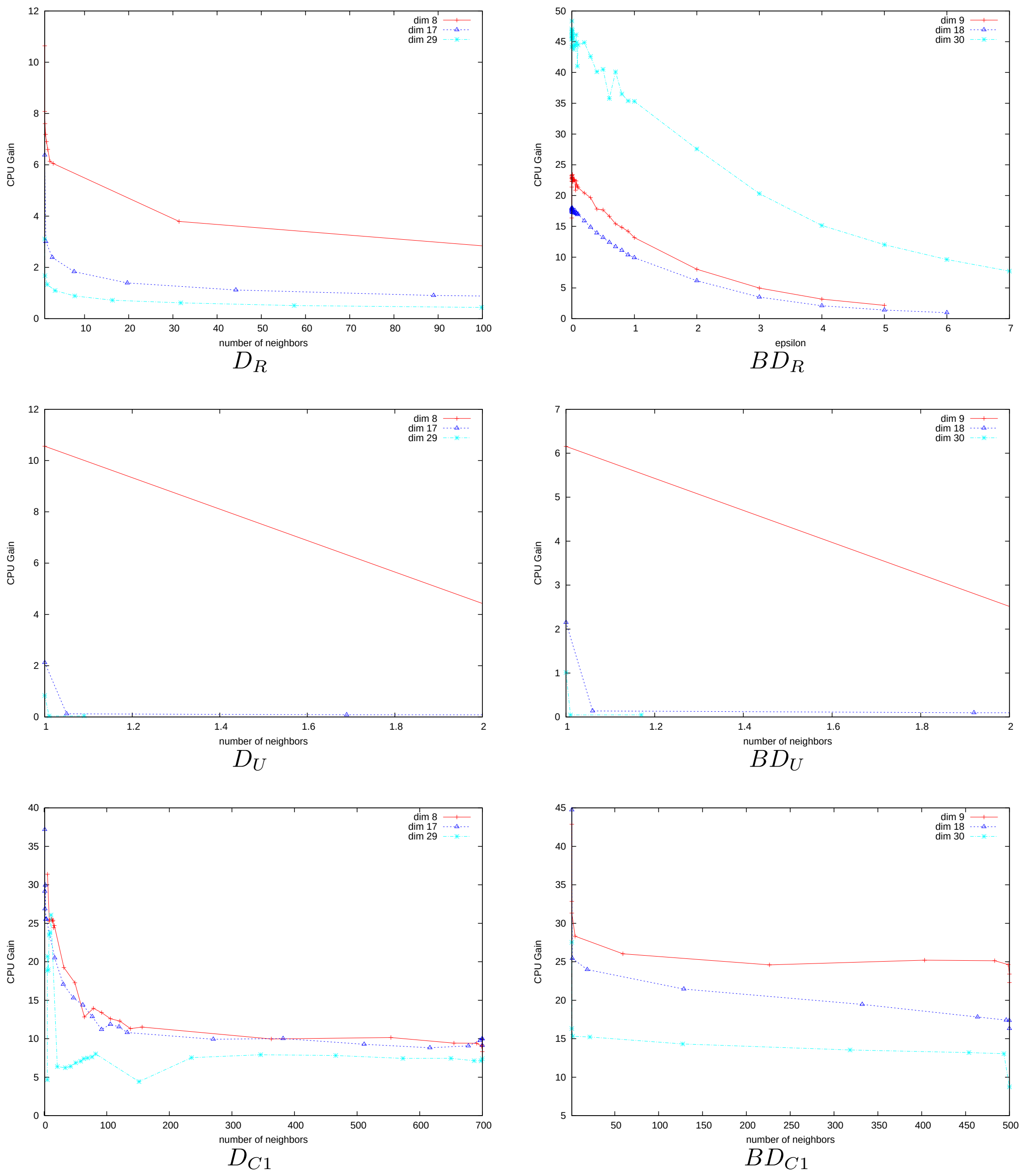


(b) Ratio of acceded nodes

Figure 10: *Zoom for the D_U distribution.*

4.4 Speed-up SR-tree/VA-File

In this section, we compare the CPU time performance of sphere queries with an SR-tree to that of a VA-File [26]. We plot in Figure 11 the speed-up SR-tree over VA-File versus the number of neighbors found except for BD_R where the speed-up versus ϵ is displayed.



(a) Small datasets (b) Big datasets

Figure 11: *CPU time speed-up for SR-tree vs VA-File.*

For the uniform distribution, as expected, even for dimension 8 or 9 the VA-File CPU time is lower than that of the SR-tree. Note however that for very large datasets, the SR-tree might outperform the VA-file not because of CPU time but of disk accesses for small dimensions.

In contrast, for clustered data as expected the SR-tree outperforms the VA-file. For the moderately clustered dataset D_R, even at dimension 17, the speed up is small (even lower than 1 for D_R in dimension 29). However due to the nature of the data, speed up reaches almost ten for the highly clustered data of BD_R with dimension 30 (the number of neighbors is around 100). With the two synthetic clustered datasets D_{C1} and BD_{C1}, the SR-tree brings one order of magnitude speed-up compared to the VA-file even for dimension 29 or 30. Recall however that the VA-file is not appropriate for non uniform distributions and that the speed-up of SR-tree wrt to more recent and more appropriate variants of VA-files such as the VA$^+$-File version [11] would probably be smaller with the evaluated datasets.

5 Conclusion

This report was devoted to the study of similarity search with sphere queries in a collection of points in a high-dimensional space. We experimentally showed that - in contrast to common ideas and expectations - when the dimension of the space is moderate and when data contains some redundancy, the use of a tree-structured index is still effective, i.e. it significantly accelerates search in comparison to sequential scan. In other words, the redundancy of data delays the impact of dimension on performance (curse of dimensionality). We have illustrated this idea with the SR-tree structure and a number of synthetic and real life datasets. From the experiments we showed, among different distributions and dimensions (up to 30 dimensions), that a tree structure is still effective for non uniform databases whereares, as expected, for the uniform data the tree structure is not efficient. We also showed that a common amelioration of sequential scan called VA-file might be outperformed by the SR-tree because of the clustering of data. This experimental study partially answers the open question: "Is concentration observed when there are dependences?" raised in [12]. The impact of this study should be stressed, since there is a large number of applications that faithfully describe (model) objects by a number of points whose dimension is only moderate. Among these applications, image and video description by interest points is noteworthy.

At the time of the experiments no real life dataset large enough so that the structures do not totally hold in central memory was available. As a future work, we intend to show that the larger the dataset, the higher the dimension where the curse of dimensionality appears. Indeed when the size increases, the impact of disk accesses on performance becomes crucial and should advantage tree-like structures.

References

[1] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger, *The R*-tree: An efficient and robust access method for points and rectangles*, Proceedings of the ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990, ACM Press, 1990, pp. 322–331.

[2] S. Belongie, J. Malik, and J. Puzicha, *Shape matching and object recognition using shape contexts*, IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002), no. 4, 509–522.

[3] Stefan Berchtold, Christian Bohm, Daniel A. Keim, Hans-Peter Kriegel, and Xiaowei Xu, *Optimal multidimensional query processing using tree striping*, Data Warehousing and Knowledge Discovery, 2000, pp. 244–257.

[4] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel, *The pyramid-technique: towards breaking the curse of dimensionality*, SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data (New York, NY, USA), ACM Press, 1998, pp. 142–153.

[5] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel, *The X-tree: An index structure for high-dimensional data*, Proceedings of the 22nd International Conference on Very Large Databases (San Francisco, U.S.A.) (T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, eds.), Morgan Kaufmann Publishers, 1996, pp. 28–39.

[6] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft, *When is "nearest neighbor" meaningful?*, Lecture Notes in Computer Science **1540** (1999), 217–235.

[7] C. Böhm, S. Berchtold, and D.A. Keim, *Searching in high-dimensional spaces - index structures for improving the performance of multimedia databases*, ACM Computing Survey **33** (2001), no. 3, 322–373.

[8] COIL-100, *The Columbia Object Image Library (100 objects)*, http://www1.cs.columbia.edu/CAVE/.

[9] M. Douze, M. Sdika, and C. Schmid, *Photomole: retrieval from a database of natural images*, Demo at the International Conference on Computer Vision, October 2005.

[10] M. Everingham, A. Zisserman, C. Williams, L. Van Gool, M. Allan, C. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorko, S. Duffner, J. Eichhorn, J. Farquhar, M. Fritz, C. Garcia, T. Griffiths, F. Jurie, D. Keysers, M. Koskela, J. Laaksonen, D. Larlus, B. Leibe, H. Meng, H. Ney, B. Schiele, C. Schmid,

E. Seemann, J. Shawe-Taylor, A. Storkey, S. Szedmak, B. Triggs, I. Ulusoy, V. Viitaniemi, and J. Zhang, *The 2005 PASCAL visual object classes challenge*, Selected Proceedings of the First PASCAL Challenges Workshop, LNAI, Springer-Verlag, 2006.

[11] Ferhatosmanoglu, Tuncel, Agrawal, and El Abbadi, *Vector approximation based indexing for non-uniform high dimensional data sets*, CIKM: ACM CIKM International Conference on Information and Knowledge Management, ACM, SIGIR, and SIGMIS, 2000.

[12] D. François, V. Wertz, and M. Verleysen, *Open questions about similarity search in high-dimensional spaces*, 23th Benelux Meeting on Systems and Control (Helvoirt, The Netherlands), 2004, p. 112.

[13] Christian Garcia-Arellano and Kenneth C. Sevcik, *Quantization techniques for similarity search in high-dimensional data spaces.*, BNCOD, 2003, pp. 75–94.

[14] V. Gouet and N. Boujemaa, *Object-based queries using color points of interest*, IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL 2001) (Kauai, Hawaii, USA), 2001, pp. 30–36.

[15] Antonin Guttman, *R-trees: A dynamic index structure for spatial searching.*, SIGMOD Conference, 1984, pp. 47–57.

[16] Douglas R. Heisterkamp and Jing Peng, *Kernel va-files for nearest neighbor search in large image databases.*

[17] J.J.Koenderink and A.J.van Doorn, *Representationof local geometry in the visual system*, Biological Cybernetics **55** (1987), 367–375.

[18] Norio Katayama and Shin'ichi Satoh, *The SR-tree: An index structure for high-dimensional nearest neighbor queries.*, SIGMOD Conference, 1997, pp. 369–380.

[19] Y. Ke and R. Sukthankar, *PCA-SIFT: A more distinctive representation for local image descriptors*, IEEE Computer Vision and Pattern Recognition **02** (2004), 506–513.

[20] David G. Lowe, *Object recognition from local scale-invariant features*, International Conference on Computer Vision (Corfu, Greece), 1999, pp. 1150–1157.

[21] Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N. Papadopoulos, and Yannis Theodoridis, *R-trees have grown everywhere*, 2003.

[22] K. Mikolajczyk and C. Schmid, *A performance evaluation of local descriptors*, IEEE Transactions on Pattern Analysis and Machine Intelligence **27** (2005), no. 10, 1615–1630.

[23] Nicolas Monne-Loccoz, *High-dimensional access methods for ecient similarity queries*, Technical Report, Centre universitaire d'informatique Computer Vision and Multimedia Laboratory, Universite de GENEVE, 2005.

[24] L. Qin, W. Zeng, W. Gao, and W. Wang, *Local invariant descriptor for image matching*, IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia, PA, USA, 2005, pp. 19–23.

[25] Bellman R., *Adaptive control processes: A guided tour*, Princeton University Press., 1961.

[26] Hans-Jörg Schek Roger Weber and Stephen Blott, *A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces.*, Very Large Data Bases, 1998, pp. 194–205.

[27] Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima, *The a-tree: An index structure for high-dimensional spaces using relative approximation*, VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 2000, pp. 516–526.

[28] C. Schmid and R. Mohr, *Local grayvalue invariants for image retrieval*, IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997), no. 5, 530–534.

[29] M. Verleysen, *Limitations and future trends in neural computation*, ch. Learning high-dimensional data, pp. 141–162, IOS Press, 2003.

[30] M. Verleysen and D. François, *The curse of dimensionality in data mining and time series prediction*, Computational Intelligence and Bioinspired Systems (F. Sandoval J. Cabestany, A. Prieto, ed.), Lecture Notes in Computer Science vol. 3512, 2005, pp. 758–770.

[31] M. Verleysen, D. François, G. Simon, and V. Wertz, *On the effects of dimensionality on data analysis with neural networks.*

[32] R. Weber and S. Blott, *An approximation based data structure for similarity search*, 1997.

[33] R. Weber, K. Bohm, and H. Schek, *Interactive-time similarity search for large image collections using parallel va-files.*

[34] Roger Weber, Hans-Jörg Schek, and Stephen Blott, *A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces.*, Very Large Data Bases, 1998, pp. 194–205.

[35] Roger Weber and Pavel Zezula, *Is similarity search useful for high dimensional spaces?*, DEXA Workshop, 1999, pp. 146–147.

[36] David A. White and Ramesh Jain, *Similarity indexing with the SS-tree*, ICDE '96: Proceedings of the Twelfth International Conference on Data Engineering (Washington, DC, USA), IEEE Computer Society, 1996, pp. 516–523.