



Content-based Retrieval Using Local Descriptors: Problems and Issues from a Database Perspective

Laurent Amsaleg and Patrick Gros

IRISA-CNRS, Campus de Beaulieu, Rennes, France

Abstract: Most existing content-based image retrieval systems built above a very large database typically compute a single descriptor per image, based for example on colour histograms. Therefore, these systems can only return images that are globally similar to the query image, but cannot return images that contain some of the objects that are in the query. Recent image processing techniques, however, focused on fine-grain image recognition to address the need of detecting similar objects in images. Fine-grain image recognition typically relies on computing many local descriptors per image. These techniques obviously increase the recognition power of retrieval systems, but also raise new problems in the design of fundamental lower-level functions such as indexes and secondary storage management. This paper addresses these problems: it shows that the three most efficient multi-dimensional indexing techniques known today do not efficiently cope with the deep changes in the retrieval process caused by the use of local descriptors. This paper also identifies several research directions to investigate before being able to build efficient image database systems supporting fine-grain recognition.

Keywords: Content-based retrieval; Dimensionality curse; Local image descriptors; Multi-dimensional indexing

1. ORIGINALITY AND CONTRIBUTIONS

Many authors coming from the image community develop new techniques to retrieve images by their contents. Their focus is on increasing the recognition power of their schemes. They usually evaluate the strength of their techniques using data sets that typically fit in main memory, therefore avoiding the secondary storage management burden. Facilitating the management of disks and removing this burden has long been a strong motivation for the database community. Because the focus of this community is not on image recognition, the very large content-based retrieval systems presented in the DB literature use simple and ordinary image processing techniques.

This paper tries to bridge this gap. We investigate the use of a very powerful image recognition technique, together with the most recent indexing schemes. That investigation has two major goals: it evaluates the performance of a recent recognition scheme at a scale larger than that typically used

in research labs, and shows the way indexes behave when used together with smart image processing techniques. Our conclusion is that current indexing schemes do not cope efficiently with powerful recognition schemes. We therefore identify several research directions to investigate before being able to build efficient, very large and powerful content-based retrieval systems.

2. INTRODUCTION

Building large content-based retrieval systems require two types of complimentary technologies to enforce fast similarity searches. First, image processing techniques are needed to extract *descriptors* from images. Descriptors are typically vectors of real numbers defining points in a high-dimensional space. They encode information found in images, and they are used during the search process. The similarity of two images is assumed to be proportional to the similarity of their descriptors, which is measured as the distance between the points defined by the descriptors. Similarity search is therefore implemented as a nearest-neighbour search or as a ϵ -range search within the feature space.

Secondly, database techniques are needed for storing

descriptors on disks, and for accelerating searches by using multi-dimensional index structures. Space-partitioning methods like grid-file [1], K-D-B-Tree [2] or LSD^h-Tree [3] divide the data space along predefined lines regardless to the actual values of data, and store each descriptor in the appropriate cell. Data-partitioning methods like R-Tree [4], X-Tree [5], SS-Tree [6], SR-Tree [7] or TV-Tree [8] divide the data space according to the distribution of data. All these access methods 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 *dimensional curse*.

Recently, two innovative approaches for indexing, the Pyramid-Tree [9] and the VA-File [29], specifically designed to tackle this phenomenon, have been published. Both provide today among the most efficient support for multi-dimensional similarity search. These techniques, however, do not cope with the requirements of fine-grain image recognition techniques based on local descriptors. Instead of computing a *single* descriptor per image (as for coarse-grain image recognition, as used in QBIC [10]), fine-grain recognition techniques compute *several* descriptors per image. To perform a search with local descriptors, the system needs to compute all the descriptors that describe the query image. Each descriptor is then used to query the index. Each (partial) answer is kept around, and once they all have been returned, they are analysed as a whole and then the final result (i.e. a set of similar images) is eventually returned. Today, with conventional image technologies (based on global descriptors) and conventional database technologies, querying a multi-dimensional index only once is barely efficient when the database of descriptors is large, and when the data space has a lot of dimensions. Techniques using local descriptors need multiple consecutive queries to get the images that are similar to a single image, and *adding-up* the response time of each individual query makes the global response time far above what one might tolerate.

This paper is an initial exploration of the consequences of using local descriptors together with up-to-date database multi-dimensional indexing strategies. We show that none of the DB techniques known today can efficiently handle similarity-search queries made of many descriptors instead of a single one, as it is assumed traditionally. We also list research directions and enumerate several potential solutions for enabling efficient content-based retrieval systems when stored images are characterised by many descriptors, as suggested by modern image processing techniques.

The remainder of this paper is structured as follows. Section 3 presents a family of local descriptors for robust object recognition in colour images. This family is a good example of recent image processing techniques. These local descriptors are therefore good candidates to evaluate the behaviour of recent database indexing techniques in a fine-grain recognition oriented context. Section 4 gives an overview of the multi-dimensional indexing techniques used in databases. Section 5 first demonstrates the recognition power of the descriptors, and then evaluates the performance of

the three most efficient techniques known today to index a large base of high-dimensional data when queries contain many descriptors. Section 6 presents some open issues and an initial set of solutions, before concluding.

3. MODERN IMAGE PROCESSING TECHNIQUES FOR CONTENT-BASED RETRIEVAL

Database and image technologies have orthogonal contributions to the construction of a complete system for content-based retrieval. Database techniques focus on the efficient management of large volumes of data and mainly address response time issues. In contrast, image techniques focus on recognition. The global recognition power of the whole system is driven by the capabilities of the image processing technique chosen. Choosing a specific type of descriptor is therefore crucial. The first part of this section motivates and presents some possible choices. This section then details an extension to colour images of the fine-grain recognition scheme for grey-level images originally proposed by Florack et al [11].

The scheme proposed by Florack et al [11] is a powerful image recognition technique. It can detect that two images contain similar objects, even if the locations of objects differ, even if they are in front of different backgrounds, seen from different viewpoints or illuminated differently. This technique is based on the computation of many local differential descriptors per image. Both its performance in terms of recognition and the way in which local descriptors are computed and used during the retrieval make this technique somehow archetypical of what a modern image processing technique could be. It is therefore interesting to investigate the behaviour of recent database indexing techniques in this context. Before presenting the way in which we extend Florack's grey descriptors to cope with colour, we briefly discuss the pros and cons of using global versus local descriptors.

3.1. Global versus Local Descriptors

Descriptors are at the root of image content-based retrieval systems. A descriptor encodes some specific information extracted from an image. Descriptors are typically represented by multi-dimensional vectors of real numbers, defining points in a multi-dimensional data space. They are usually designed so that the points they define in the data space are close if the images they are associated with are similar. Depending on the computation strategy of the descriptors, images may be found similar despite illumination changes, viewpoint variations, or partial occlusions. Families of descriptors are defined with respect to the type of variations they are invariant to. Many techniques have been designed to make descriptors *invariant* to a large collection of variations between images [12–18].

Traditionally, a descriptor encodes information that is *global* to an image. In this case, the system typically computes

a single descriptor per image. Colour histograms, grey-level histograms or correlograms are typical examples [19,20]. It has been demonstrated that global descriptors are robust, i.e. they are little affected by the noise of the digital signal. They cannot be used, however, to detect similar elements between images like regions or objects, especially if the backgrounds behind objects differ and occupy the major part of the images.

To answer the increasing need to identify elements within images, more recent image processing techniques have designed recognition schemes based on *local descriptors*. This fine-grain image recognition (as opposed to global descriptors that enable large-grain recognitions) is extremely powerful. With local descriptors, it is, for example, possible to identify an object independently of its position in the image, or even if it is partially occluded, whereas this is very hazardous when using global descriptors.¹ Computing local descriptors is expensive because many of them are derived from a single image, typically between 50 and 600. This extra cost is by far amortised by their increased recognition power.

Using local descriptors instead of global descriptors dramatically changes the computational chain when searching for similar images. First, it increases the size of the database: instead of storing one descriptor per image, many of them must be kept for each image. The size of the database is typically increased by two orders of magnitude. Secondly, it changes the way similar images are searched: instead of searching the descriptors that are close to the unique query descriptor (as is the case with global descriptors), the database must be queried many times, each time using a different query (local) descriptor, each partial result must be post-processed and the similar images are known only once all the query descriptors have been used. Therefore, searching for images that are similar to a given one typically generates between 50 and 600 consecutive queries which search in a database that is 50 to 600 times larger. This demanding process increases the impact of the performance problems traditional multi-dimensional index techniques suffer from.

3.2. The Local Differential Descriptors Family

The descriptors we now present are an extension to colour images of the fine-grain recognition scheme for grey-level images originally proposed by Florack et al [11], and extensively used and evaluated by Schmid and Mohr [21]. We built on this scheme because it is highly robust to grey-level image transformations: it detects similar elements in images despite orientation changes (rotations), translations, resolution changes, illumination variations, partial occlusions, changes of backgrounds or viewpoints, etc. Coping with colour images instead of grey-level images has a deep impact on the way in which the descriptors handle (i.e. absorb) the variations of illumination. Before going into detail, the scheme proposed by Florack et al [11] can be roughly outlined as follows.

Computing the local descriptors that encode information about a single image is done in three steps. First, specific points in the image, called interest points, are selected. The number of points in one image varies, since it depends on the shape of the signal of that image. Secondly, the signal around each interest point is characterised by its convolution with a Gaussian function and its derivatives up to the third order. Thirdly, these derivatives are mixed to enforce invariance properties, and to make descriptors robust to the changes mentioned above. A descriptor is typically a vector of real numbers having seven or nine dimensions.

Descriptors are then inserted into an index. To know which image a descriptor has been computed from, image identifiers are stored together with the descriptors. Computing descriptors over all the images is done off-line. The similarity retrieval proceeds as follows: interest points are first identified in the query image, and the corresponding local descriptors are then computed. Each descriptor of the query is used to probe the index. The index returns similar descriptors found in the database (with respect to a nearest-neighbour or a ϵ -search), from which it is possible to determine the id of the associated image. It is therefore easy to count the number of times each image id is returned by the index during the whole retrieval process. At the end of this process (i.e. once all the local descriptors of the query have been used to probe the index), the counters are used to rank the candidate images by decreasing similarity.

We now present in more detail the computation process of our descriptors.

3.2.1. Extracting Interest Points. Interest points are determined so that it is very likely that a point found in one image will be also found in another image which slightly differs from the first one. Schmid [22], who extensively used Florack's method, compared several point extractors, and showed that the extractor introduced by Bigün, Granlund and Wiklund [23], and improved by Harris and Stephens [24], had the best behaviour.

This extractor looks for 2D singularities in the signal, and is based on the computation of the eigenvalues of the matrix

$$e^{-\frac{x^2 + y^2}{2\sigma^2}} \otimes \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

where I_x and I_y are the convolution of the signal with the two derivatives $\partial G/\partial x$ and $\partial G/\partial y$ of a Gaussian function. In general, there are many points in each image, typically between 50 and 600.

Dufournaud, Schmid and Horaud [25] pointed-out that descriptors could not be invariant to scale factors if the points extracted were not invariant to scaling. Therefore, he proposed to extract the points at different scales by varying the variance of the Gaussian function used. This allow us to cope with scale factors up to 7, rather than up to 2 as for the original method.

¹ Although this may change, since 'partial queries' is an active field of research.

3.2.2. Computing Local Descriptors for Grey-Level Images. Once the points are extracted, the descriptors are computed using the signal around each of the points. This computation is done in two steps. During the first step, the signal is convoluted with a Gaussian function and its first nine derivatives (the derivatives up to the third order). These smoothed derivatives provide a basic description of the signal. During the second step, the derivatives are mixed together to enforce invariance properties, and to make descriptors robust to the changes mentioned above. We describe below the way in which these derivatives are mixed.

- *Gaining Translational and Rotational Invariance.* Translational invariance is obtained by the fact that the descriptors are computed around each interest point.

The angle of rotation of images can be algebraically eliminated from the ten derivatives, providing nine resulting quantities invariant to rotations. If the smoothed signal is denoted by I , and its derivatives by I_x, I_y, I_{xx}, \dots , these nine invariant quantities are:

$$\begin{aligned}
 & I \\
 & I_x I_x \\
 & I_x I_y \\
 & I_{xx} \\
 & I_{xy} \\
 & I_{yy} \\
 & I_{xx} I_{xx} - I_{xy} I_{xy} \\
 & I_{xy} I_{xy} - I_{yy} I_{yy} \\
 & I_{xx} I_{yy} - I_{xy} I_{xy}
 \end{aligned} \tag{1}$$

The function ϵ_{ij} is defined by $\epsilon_{xy} = -\epsilon_{yx} = 1$, $\epsilon_{xx} = \epsilon_{yy} = 0$. Einstein's notation used in the formulas corresponds to a summation over each index: for instance, $I_i = \sum_i I_i = I_x + I_y$ and $I_{ij} I_{ji} = \sum_i \sum_j I_{ij} I_{ji} = I_{xx} I_{xx} + 2I_{xy} I_{xy} + I_{yy} I_{yy}$.

- *Gaining Photometric Invariance.* The descriptors are invariant to the variations of illumination that are modelled by $I \rightarrow aI + b$. This model, although simple, describes quite accurately what happens when the global intensity of the illumination varies slightly. It is possible to withdraw these two new parameters a and b and to obtain seven rotational and photometric invariants. First, withdraw the two first quantities of Eq. (1) and divide each of the seven last ones by the appropriate power of I_x , so as to obtain ratios of degree 0 with respect to I and its derivatives.
- *Gaining Scale Invariance.* Invariance to scale is achieved by adopting a multi-scale approach: the computation of invariants is repeated for various values of the variance of the Gaussian, and all the resulting values are used to describe the image. The values of the variance used here

are related to those used during the extraction of the interest points: the variance used to extract a point gives the variance to compute the associated descriptor.

3.2.3. Extension to Colour Images. The method used to compute the local descriptors for grey-level images is very robust, as evaluated in Schmid and Mohr [21]. We therefore extended this method to cope with colour images. Each pixel of a colour image is defined by three values, which can be coded in many ways [26]: RGB, HSV, Lab, etc. The RGB system is chosen because it facilitates the extension of the descriptors to colour images.

Extending the local descriptors does not change the way in which the interest points are extracted. We still use Harris's detector. The signal, however, is now characterised by 30 derivatives (10 per channel). Coping with colour significantly changes the way in which these derivatives must be mixed to gain invariance towards rotation and illumination variations. We detail these changes below.

- *Gaining Rotational Invariance.* Rotational invariance is obtained by withdrawing the angle of rotation. 3×9 invariants can be computed with Eq. (1) applied on each channel, and two others are chosen from among the three following quantities (for numerical reasons, it is wise to keep all three values in practice):

$$R_x G_x + R_y G_y \quad R_x B_x + R_y B_y \quad G_x B_x + G_y B_y \tag{2}$$

Scale invariance is obtained by a multi-scale approach similar to that used with grey-level images.

- *Gaining Photometric Invariance.* Photometric invariance is more complex with colours because different illumination models can be considered. A general model is $(R', G', B')^T = M(R, G, B)^T + V$, where M is a 3×3 matrix and V a vector of dimension 3. Other models are obtained by using a diagonal or scalar matrix for M , and by possibly removing the vector V . According to the number of parameters of the model, the dimension of the descriptors varies between 18 and 29.

A very common model is obtained when M is diagonal. In this case, the three channels remain independent when the image is transformed. The descriptors can thus be computed on each channel as they were in the grey-level case, and by adding two more dimensions using the formulas in Eq. (2). This provides descriptors of dimension 24.

Another case is that of a full rank M matrix when no rotational invariance is needed. In this case, the vectors (R, G, B) , (R_x, G_x, B_x) , $(R_y, G_y, B_y) \dots$ are all subjects to the same linear or affine transform. The vector V can be withdrawn by not considering the vector (R, G, B) . The linear part remains. The basic invariants in this case are the coordinates of the vectors with respect to three of them chosen as a reference frame. If we choose (for reasons of symmetry) the three vectors (R_{xx}, G_{xx}, B_{xx}) , (R_{xy}, G_{xy}, B_{xy}) and (R_{yy}, G_{yy}, B_{yy}) as a reference frame, then the invariants are:

$$\begin{array}{c} \left| \begin{array}{ccc} X & R_{xy} & R_{yy} \\ Y & G_{xy} & G_{yy} \\ Z & B_{xy} & B_{yy} \end{array} \right|, \left| \begin{array}{ccc} R_{xx} & X & R_{yy} \\ G_{xx} & Y & G_{yy} \\ B_{xx} & Z & B_{yy} \end{array} \right|, \left| \begin{array}{ccc} R_{xx} & R_{xy} & X \\ G_{xx} & G_{xy} & Y \\ B_{xx} & B_{xy} & Z \end{array} \right|, \dots \\ \left| \begin{array}{ccc} R_{xx} & R_{xy} & R_{yy} \\ G_{xx} & G_{xy} & G_{yy} \\ B_{xx} & B_{xy} & B_{yy} \end{array} \right|, \left| \begin{array}{ccc} R_{xx} & R_{xy} & R_{yy} \\ G_{xx} & G_{xy} & G_{yy} \\ B_{xx} & B_{xy} & B_{yy} \end{array} \right|, \left| \begin{array}{ccc} R_{xx} & R_{xy} & R_{yy} \\ G_{xx} & G_{xy} & G_{yy} \\ B_{xx} & B_{xy} & B_{yy} \end{array} \right| \end{array}$$

where (X, Y, Z) is respectively equal to (R_x, G_x, B_x) , (R_y, G_y, B_y) , $(R_{xxx}, G_{xxx}, B_{xxx}) \dots$. The invariants to both rotations and a photometric model with full rank matrix M have not been derived yet, as far as we know.

4. DATABASE TECHNIQUES FOR INDEXING NUMEROUS IMAGES

This section gives an overview of the techniques used in databases for indexing multimedia data (often focusing on still images). Database indexing techniques are needed as soon as the space required to store all the descriptors gets too big to fit in the main memory. Say, for example, that each image is described by 100 descriptors having 24 dimensions. In this case, 9 Gb of storage per million of images is needed to store the associated descriptors, and this strongly suggests keeping them on disks. Database indexing techniques are therefore used for storing descriptors on disks and for accelerating the search process by using multi-dimensional index structures. Their goal is to minimise the resulting number of I/Os. In the following, we first present the traditional approaches used for multimedia indexing from a database perspective. We then detail the two strategies that today provide the most efficient support for multi-dimensional searches.

4.1. Traditional Approaches

Database multimedia indexing techniques can be classified into two families: *data-partitioning index methods* that divide the data space according to the distribution of data; and *space-partitioning index methods* that divide the data space along predefined lines, regardless to the actual values of data, and store each descriptor in the appropriate cell. All techniques fill the data space with descriptors, or with approximations of them.

Data-partitioning index methods all derive from the seminal R-Tree [4], originally designed for indexing bi-dimensional data used in Geographical Information Systems. Minimum bounding rectangles and overlapping are the key concepts of this technique: the leaves of the tree reference each object through its minimum bounding rectangle, and the internal levels of the tree store the rectangles that overlap and bound the rectangles of a lower level. Various strategies can be used to determine which rectangles should be merged or kept separated at each level of the tree [5].

The R-tree was later extended to cope with multi-dimensional data. The SS-Tree [6] is an extension that relies on spheres instead of rectangles. While spheres improve the performance of the search process, it has been demonstrated that bounding spheres occupy a much larger volume than bounding rectangles with high-dimensional data, and that this reduces the search efficiency. Therefore, to overcome this drawback, the SR-Tree [7] specifies its regions as the intersection of a bounding sphere and a bounding rectangle.

The TV-Tree [8] is another extension of the R-Tree. The underlying idea of this approach comes from the observation that not all dimensions play the same role during the search: some are more discriminative than others. Since the search performance is partly determined by the number of dimensions of the data, Lin, Jagadish and Faloutsos [8] divide the dimensions into three classes: the dimensions that the search process can always ignore; those always used; and those that may be used to refine the search. The major drawback of this technique is that it requires an accurate knowledge of the distribution of data along each dimension as a prerequisite.

Tree construction is usually achieved (this applies to all the approaches mentioned above) by splitting nodes in overflow into two equally filled nodes, i.e. at the 50%-quantile. This fill factor enforces balanced trees and maximises disk usage. However, Berchtold et al [9] demonstrate that in the general case, using a 50%-quantile leads to the unexpected effect that, in high-dimensional spaces, the probability of accessing every index page gets close to 1. It is therefore likely that the whole index has to be scanned during a search process. The resulting access pattern to disk pages severely hampers the search performance, since it is totally random. Consequently, a traditional sequential search that scans the whole database is often a faster process for similarity search.

Space-partitioning techniques like grid-file [1], K-D-B-Tree [2] and the LSD^b-Tree [3] typically divide the data space along predetermined lines regardless of data clusters. Actual data are subsequently stored in the appropriate cells. These techniques are known to become inefficient when the dimension of data increases: above 10–16 dimensions, a simple linear scan of the entire database is typically faster than all of these techniques. They also face the problem of indexing large volumes of empty space. For example, dividing each of the 30 dimensions of a data space into two distinct regions creates 2^{30} cells. This number is by far greater than the typical number of points filling the data space. In addition, when the query point is near a cell boundary, the search process may have to lookup many cells in the neighbourhood, increasing the search cost. Furthermore, it is likely that most of these cells are empty. The evaluation of the most recent approaches [27,28] shows that they are efficient with low-dimensions and for a small amount of noise (ϵ -search).

4.2. VA-File and Pyramid-Tree

All the techniques presented above generally work well for low-dimensional spaces. Their performance, however, is

known to degrade as the number of dimensions of the descriptors increase, e.g. above 10–16, as evaluated by Weber et al [29]. This phenomenon is known as the *dimensional curse*.

In other words, any navigation within the index structure becomes more costly than a simple sequential scan in high-dimension spaces. Two innovative approaches, the Pyramid-Tree [9] and the VA-File [29], however, have recently been proposed to tackle the dimensional curse phenomenon head-on: they have been designed specifically such that their behaviour does not dramatically degenerate when the indexed data has many dimensions. These two strategies today provide among the most efficient support for multi-dimensional similarity search. We therefore used them for our performance evaluations (see Section 5).

With the VA-File, Weber et al [29] proposed a method that improves the performance of the (simple) sequential scan, since this technique proves to be competitive in high-dimensions. Their method manages two different sets of data: a file storing all the descriptors; and another file storing the geometrical approximations of these descriptors. The performance of this method is at its best when this latter file fits in main memory.

To compute the geometrical approximation of the descriptors, the method first split each dimension d_i in 2^{b_i} slices (coded using b_i bits), such that all slices are equally full. All d dimensions are sliced in this way. The intersection of slices define 2^b cells, where $b = \sum b_i$, numbered from 0 to $2^b - 1$. To fill the index, all the descriptors are then read, and the approximation of a descriptor is given by the cell number into which it falls. The file storing the geometrical approximations of the descriptors therefore associates a descriptor id to a cell number. Only cells in which at least one descriptor fall are kept in the file, avoiding the problem of managing many empty cells, as mentioned above.

During a search, the query descriptor is processed in a similar manner. The algorithm first computes its geometrical approximation, determines which cells are close to the query cell, and scans them in an increasing order of distance. Starting from the closest cell, the algorithm sequentially fetches the associated descriptors and performs distance calculations. This process is repeated until n nearest-neighbours are found. Restricting the search to close cells and ordering their investigation filters out cells that may not be part of the result, and therefore filters out all the irrelevant descriptors. This reduces the number of records to fetch and the number of comparisons and calculations to perform with respect to the traditional sequential scan.

Berchtold et al [9] proposed, with the Pyramid-Tree, a method that divides a space $[0,1]^d$ into $2 \times d$ pyramids. The top of each pyramid is placed at the centre of the data space $(0.5, \dots, 0.5)$. The base of each pyramid has a surface of $d - 1$ dimensions. Each pyramid is assigned a different number. Each pyramid is then cut into slices that are parallel to its base. The nature of pyramids is such that the slices close to their top are smaller than those near their base. This division of the data space has the interesting property to create a number of cells that increase linearly (and not exponentially) with the number of dimensions.

Dividing the data space into sliced-pyramids enables them to map any point of the multi-dimensional space into a pair (pyramid number, height in the pyramid). Because of this mapping, a B⁺-Tree index can be used instead of a multi-dimensional index structure. B⁺-Trees are known to be very efficient for this type of data and for range queries. A given slice of a specific pyramid is stored as a page of the B⁺-Tree. In addition to their efficiency, B⁺-Trees are known to cope well with concurrent updates, and can be made failure resistant. These two properties are very desirable, and often lack to other solutions.

5. PERFORMANCE EVALUATIONS

This section summarises the evaluation of the recognition power of our extension to colour images of Florack's descriptors and the performance of the VA-File, the Pyramid-Tree and the sequential scan when used together with these descriptors. As mentioned earlier, these three database techniques proved to be efficient in the context of nearest-neighbour or ϵ -range searches with global descriptors within a multi-dimensional space. We therefore measured their performance when used together with local descriptors. A first experiment compares the recognition power of colour and grey descriptors. The second experiment shows the performance of the search techniques when the dimension of the local descriptors increases. The third experiment shows the impact of the size of the database on the response times. Finally, the fourth experiment, which is the most relevant to this paper, shows the influence of the (large) number of descriptors forming a single query on the response times. We first describe our experimental setup.

5.1. Experimental Environment

To perform our performance evaluations² we used the source code of the VA-File and of the Pyramid-Tree provided by their respective authors. We also implemented our own version of the sequential search. All the algorithms were run on a SUN Ultra 5 workstation running SunOS 5.7. Its CPU is a 333 MHz UltraSPARC-IIi, with 384 Mb of main memory and 8 Gb of local secondary storage. All the response times reported here have been obtained using `getrusage()`.

We analysed the codes of the VA-File and of the Pyramid-Tree to insert at the appropriate places timer start and stop instructions. We slightly changed the metric used by the Pyramid-Tree to compute the distances between points in the data space: it was L_∞ and we changed it to L_2 . Without this patch, the nearest-neighbours returned by the Pyramid-Tree would not have been identical to those returned both by the VA-file and the sequential search. This patch seems to have no impact of the response-time.

² We are grateful to Roger Weber who graciously gave us his implementation of the VA-File. The source code of the Pyramid-Tree is available on the Web page of Stefan Berchtold (<http://www.stb-gmbh.de/~berchtol/>).

In addition, we implemented in C++ a sequential search strategy.

5.2. Overview of the Database

For the different experiments shown below, three databases were created. The first one is made of 24-dimension local descriptors derived from 1816 real-life colour images. 1206 images come from 50 seconds of a video. The remaining images come from a database of still images.³ The total number of descriptors computed from these images is 413,412, and the database therefore occupies about 40 Mb on disk.

The distribution of the descriptors along each dimension is far from being uniform. For example, the second component varies from about -20 to about 36 , and 99.14% of the values are between -1 and $+1$. Since many performance evaluations published in the literature assume uniformity of data distribution, we generated another database in which the $24 \times 413,412$ values have been picked between 0 and 1 using a random uniform generator.

As the colour descriptors have never been compared to grey descriptors, as far as we know, the 1806 images were coded in grey levels, and the corresponding seven dimension grey descriptors were computed. This third base is used in the recognition evaluation test only.

5.3 Experiment 1: Comparing the Recognition Power of Colour and Grey Descriptors

Before evaluating the performance of the different indexing algorithms, it is important to verify the recognition power of the descriptors. Some results have already been published using grey level descriptors [30]. Therefore, this experiment focuses on a comparison between the (original) 7D grey level descriptors and our 24D colour descriptors. The aim is to decide whether the colour descriptors are worth their additional complexity in terms of size and computation.

Two databases derived from the 1806 real-life images were created for this experiment. The first database keeps the 24D descriptors computed from the images. The second database stores the 7D descriptors computed from the same images. Both databases contain exactly 413,412 descriptors. Their sizes are different, however. The database for 24D descriptors is of 39,687,552 bytes, whereas the database keeping 7D descriptors is of 11,575,536 bytes.

A first set of tests was done by querying the database with images coming from sequences presenting a specific variation to stress the robustness of the descriptors towards this variation. For each of these tests, we show, in all the figures below, the query image, the closest images retrieved according to the descriptors, and possibly some false negative, i.e. images belonging to the same sequence as the one the query image belongs to, but that were not, however, retrieved by the system.

Under each query image is indicated the number of

descriptors computed for this image. This number is always greater for grey-level descriptors than for colour descriptors, since it is always possible to compute, in the grey-level case, a descriptor for each interest point extracted. On the other hand, some of these points correspond to the similarity of only one or two of the three colour channels, and can therefore not be used to compute 24D descriptors. The number of descriptors that matched is indicated together with each image returned by the system. For those images retrieved using both 24D and 7D images, a label is also indicated to allow a precise check of the results without ambiguities.

In order to obtain results at a larger scale, all the images from the video were used as queries in a second set of tests.

5.3.1. Specific Tests of Robustness.

- *Evaluation of Invariance towards Illumination Intensity Variations.* This first experiment tests the robustness of the descriptors towards the intensity of the main light source. Such a variation is very common, and all descriptors should be robust to it. Figure 1 shows the results obtained with our descriptors. The results obtained with 7D and 24D descriptors are similar. The only difference is a better difference of scores between the relevant images and the first non-relevant image when the 24D descriptors are used. This would probably be a problem with larger databases when using the 7D descriptors.

The false negative can be explained by the fact that this image is quite saturated, a phenomenon which has not been taken into account when the descriptors were forged.

- *Evaluation of invariance towards spectrum variations.* This second experiment is concerned with variations in the spectrum of the main light source. Although such variations are common at small scale (for example, the morning sunlight is different from the evening sunlight), they are not easy to produce experimentally. We used a set of colour filters placed in front of our main light source, but these filters are usually too colourful to simulate natural phenomena. On the other hand, this allows us to test the descriptors in rather extreme cases.

The results of Fig. 2 show a better performance of 7D descriptors. This is not surprising, since they mix the three channels together and can compensate for the fact that only one of the three channels is really useful in some of the images. On the other hand, the 24D descriptors assume that the different channels are independent, and this assumption is clearly wrong in the present case.

- *Evaluation of invariance towards the motion of the main light source.* The last photometric variation we tested is that arising when the light source is moving around the scene. If the results shown in Fig. 3 are similar in terms of which images are retrieved, the difference of score between relevant and irrelevant images is much smaller with 7D descriptors than it is with 24D descriptors. This expresses a smaller power of discrimination of the 7D descriptors.

³ <http://www.inrialpes.fr/movi/pub/Images/index.html>






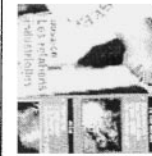



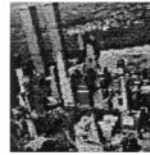










Results using 24D descriptors					
Query	Answers				False Neg.
					
518	A_1 : 346	A_2 : 341	A_3 : 307	A_4 : 259	B_1
					
	A_5 : 222	A_6 : 115	A_7 : 71	26	
Results using 7D descriptors					
Query	Answers				False Neg.
					
572	A_2 : 206	A_1 : 184	A_3 : 138	A_4 : 126	B_1
					
	A_5 : 103	A_6 : 71	A_7 : 43	27	

Fig. 1. Robustness towards variation of illumination. An example with the query image, the eight most similar images retrieved using colour and grey descriptors, and false negatives.

- *Evaluation of Geometric Invariance.* Rather than testing every possible simple motion of the camera individually, we chose a sequence of images where the camera has a complex motion of rotation and translation around the scene. The results of Fig. 4 show much better results for 24D descriptors, both in terms of discrimination and retrieval.
- *Evaluation of Composition Invariance.* This test shows the robustness of the descriptors when the composition of the scene varies. More objects cause occlusions, for example. It is clear that adding objects provides better results than removing some of them, since removed objects are the cause of less interest points and less descriptors in the image. The results of Fig. 5 are slightly better with 24D descriptors. As was the case in the previous tests, these descriptors have also a better discrimination power.
- *Evaluation with real life images.* This test was not designed to test a specific variation, but rather to test the behaviour of the descriptors with real life images. The sequence used presents an actress moving: this is typically a case of complex 3D motion. As is the case with geometric invariance, 24D descriptors provide much better results (see Fig. 6.)

Conclusion. It could seem surprising that the 24D descriptors are much more robust to geometric transformations of the image than the 7D grey level descriptors, and that the difference in performance is much smaller with respect to photometric variations. This is because the colour illumination model used is more specific and therefore more restrictive than the model for grey-level images. On the other hand, the 24D space being much bigger, the descrip-










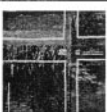







Results using 24D descriptors				
Query	Answers			False Neg.
				
408	$A_1: 296$	$A_2: 165$	$A_3: 69$	B_1
				
	19	19	$A_4: 18$	B_2
Results using 7D descriptors				
Query	Answers			False Neg.
				
417	$A_1: 256$	$A_3: 209$	$B_1: 151$	B_2
				
	$A_2: 65$	$A_4: 19$	16	

Fig. 2. Robustness towards light spectrum variations. An example with the query image, the six most similar images retrieved using colour and grey descriptors, and false negatives.

tors associated with irrelevant images are usually further and cause less noise than in the smaller 7D descriptor space.

5.3.2. Statistical Results. To obtain results at a larger scale, another experiment was done by using the 1206 images from the video, and we tested the ability of the grey and colour descriptors to retrieve from their respective databases all the images of a shot given one image of this shot as a query. This is certainly a biased definition of what similarity means in general, but it provides an unambiguous way to measure the results. The video sequence contains 38 shots without any special transition effects (fade-off, wipes, etc.).

From a practical point of view, each image of the video is used as a query. The system computes the descriptors of this image and retrieves the 10 nearest neighbours of each one of these descriptors in the corresponding database. Each of these neighbours gives a vote to the image it belongs to. Since each image has an average of 290 descriptors, it generate around 2900 votes. Only the 15 images which obtained the greatest number of votes are retained: these images (called nearest images in the latter) can be considered as the nearest neighbours of the query image.

We define the score of each query as the rank of the furthest of the nearest images which belongs to the shot of the query image, and such that all nearer images also belong to this same shot. If this score is 15, it means that all the 15 nearest images retained belong to the same shot. On the



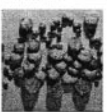















Results using 24D descriptors					
Query	Answers				
					
447	$A_1: 86$	$A_2: 62$	$A_3: 61$	$A_4: 60$	
					
	$A_5: 58$	$A_6: 47$	20		
Results using 7D descriptors					
Query	Answers				
					
456	$A_1: 47$	$A_4: 34$	$A_3: 31$	$A_5: 28$	
					
	$A_6: 24$	$A_2: 20$	18		

Fig. 3. Robustness towards light source motion. An example with the query image and the seven most similar images retrieved using colour and grey descriptors.

other hand, if this score is just 1, it means that only the first nearest neighbour (i.e. the image itself, in our experience) belongs to the same shot, the second nearest image being from elsewhere. In this latter case, the recognition process totally fails.

Once all the queries have been processed, the scores obtained are histogrammed, and the results are shown in Fig. 7. The third curve is the ground truth that was obtained by counting the number of images of each shot manually (therefore the curve shows how many images belong to a shot whose length is 8, 9... 14 or more than 15 images).

These histograms strongly depend upon the number of static shots in the video, which are the easiest ones in terms of recognition. On the other hand, their comparison demonstrates the interest in using the more complex but also more powerful colour descriptors.

The mean of the ground truth histogram is 14.77. The mean of the histogram obtained with colour descriptors is 12.73, and 9.87 for the grey descriptors. This means that three more images were correctly classified as nearest neighbours using the colour descriptors than with the grey descriptors. Grey descriptors provide better results for five images, and colour descriptors do best for 663 of them.

The difference in performance between the two types of descriptors is due to three kinds of shot: close-up on moving persons; dark images; and fast lateral travelling motions, or highly moving shots more generally. For example, the video



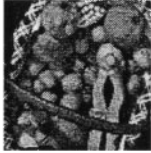




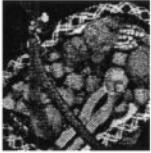




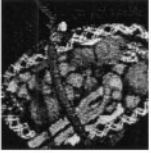












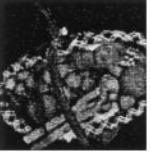


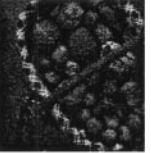

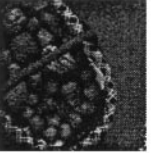

Results using 24D descriptors								
Query	Answers							
 255	 $A_1: 72$	 $A_2: 57$	 $A_3: 52$	 $A_4: 37$	 $A_5: 37$			
	 36	 $A_6: 29$	 $A_7: 28$	 $A_8: 26$	 25			
	 $A_9: 22$	 18	 17	 17	 16			
	Results using 7D descriptors							
	Query	Answers						
	 272	 $A_1: 39$	 $A_2: 30$	 $A_3: 27$	 $A_7: 27$	 $A_4: 17$		
		 $A_8: 16$	 $A_6: 14$	 $A_5: 14$	 $A_9: 13$	 13		
		 13	 12	 12	 11	 11		

Fig. 4. Robustness towards geometric variations. An example with the query image and the 15 most similar images retrieved using colour and grey descriptors.









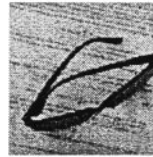





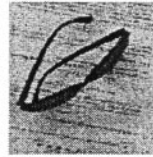


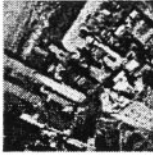

Results using 24D descriptors						
Query	Answers					
						
334	$A_1: 247$	$A_2: 166$	$A_3: 153$	$A_4: 108$		
						
	$A_5: 63$	$A_6: 31$	$A_7: 26$	21		
Results using 7D descriptors						
Query	Answers					False Neg.
						
350	$A_1: 143$	$A_2: 70$	$A_4: 55$	$A_3: 53$		
						
	$A_5: 30$	$A_6: 16$	16	$A_7: 14$		

Fig. 5. Robustness towards variations in the composition of images. An example with a query image and the eight most similar images retrieved using colour and grey descriptors.

used contains several shots taken from a helicopter following rapidly moving cars.

5.4. Experiment 2: Influence of the Dimensionality of Data

The second experiment shows the influence of the dimensionality of data on the performance of the three DB techniques we study here. For this experiment, we first computed the 413,412 descriptors having 24 dimensions using real data from the database described above. These descriptors were then truncated to 2, 4, 7, 10, 15, 20 and 24 dimensions. The other dimensions are used to get intermediate results. 413,412 descriptors following a uniform distribution have then been randomly generated for the same dimensions, but also for greater dimensions (up to

1000) for experimental purposes. The sizes of the resulting databases are given by Table 1.

Once the databases have been created, a query containing 150 descriptors was computed using an image outside the database, or new random numbers. We then truncated them to the appropriate dimensions in order to create the requests that will query the real and synthetic databases. The response times given in Fig 8 and 9 are the cumulative response times of 150 consecutive databases interrogations, each returning 10 nearest neighbours.

The performance of the algorithms using real data is illustrated by Fig. 8. In this case, the performance of the Pyramid-Tree severely degrades above seven dimensions. Beyond that, the response time of this technique is too big to remain competitive. The VA-File and the sequential search clearly exhibit better performance, and degrade less





























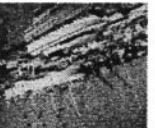
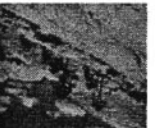
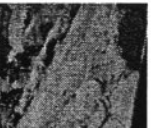

Results using 24D descriptors					
Query	Answers				Some False Neg.
 169	 A ₁ : 47	 A ₂ : 44	 A ₃ : 40	 A ₄ : 24	  
	 A ₅ : 22	 A ₆ : 16	 A ₇ : 15	 10	
	 10	 9	 A ₈ : 9	 8	
Results using 7D descriptors					
Query	Answers				Some False Neg.
 169	 A ₁ : 18	 A ₂ : 17	 A ₃ : 15	 A ₄ : 12	 A ₅  A ₇  A ₈
	 9	 9	 8	 7	
	 7	 7	 7	 A ₆ : 7	

Fig. 6. Querying with an image extracted from a video: the most similar images come from the same shot.

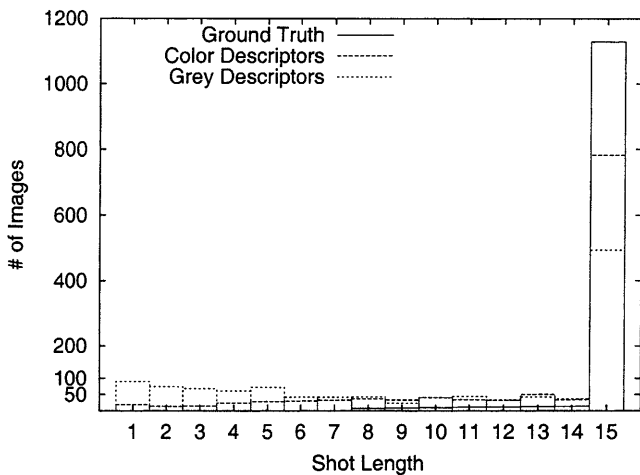


Fig. 7. Scores histograms for colour and grey descriptors and ground truth of the test.

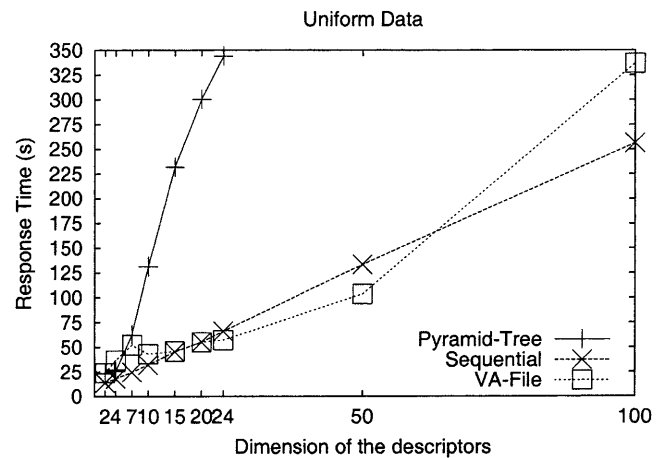


Fig. 9. Uniform database storing 413,412 descriptors, 150 descriptors in each query, increasing dimension of descriptors.

Table 1. Size (in bytes) of the databases for various dimensions of the 413,412 descriptors stored

Dimension	Size	Dimension	Size	Dimension	Size
2	3,307,296	15	24,804,720	100	165,364,800
4	6,614,592	20	33,072,960	250	413,412,000
7	11,575,536	24	39,687,552	500	826,824,000
10	16,536,480	50	82,682,400	1000	1,653,648,000

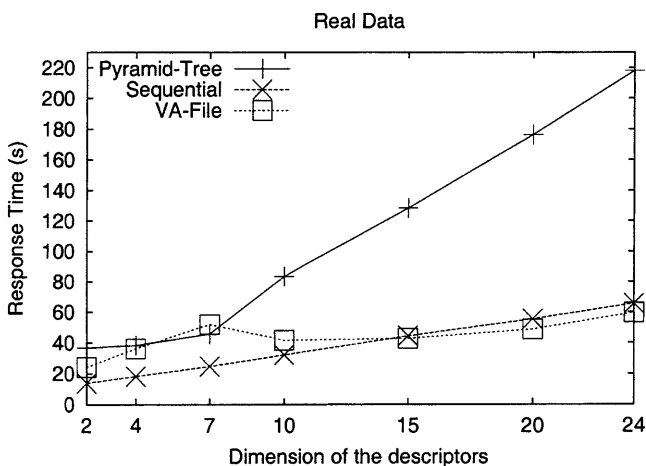


Fig. 8. Real database storing 413,412 descriptors, 150 descriptors in each query, increasing dimension of descriptors.

rapidly when the dimension of the data increases. The performance of the sequential search is linear with the dimension, which is normal and without surprises. Sequentially searching 150 descriptors among 413,412 descriptors takes approximately 14 seconds in a 2-dimensional data space, and about 66 seconds for 24 dimensions.

The performance of the VA-File and the sequential search are similar, except when the number of dimensions is small.

In this case, for two dimensions, a VA-File search takes about 24 seconds, and about 52 seconds in seven dimensions (25 seconds are needed in 7 dimensions for the sequential search). When the number of dimensions grows, the VA-File can divide its data space into smaller cells, thereby augmenting the efficiency of its filtering strategy. On the other hand, fewer dimensions makes the filtering less selective, and exploiting the approximations in addition to computing many actual distances is part of the observed overhead.

The performance corresponding to the experiments that use uniform data is given by Fig. 9. This figure does not show any response time of searches for data having more than 100 dimensions. Above that level, the VA-File never returned any answer, probably due to a bug. We did not measure the response times for the Pyramid-Tree beyond 24 dimensions, since it becomes too high to remain significant.

In this figure, the Pyramid-Tree is again the technique having the worst response time. Below 15 dimensions, the sequential search performs better than the VA-File, for similar reasons as those mentioned above. When data has 50 dimensions, the VA-File returns its answer (recall that 150 consecutive queries must be submitted before returning the answer) in about 104 seconds, while the sequential search needs 134 seconds. In this case, the VA-File strongly benefits from the uniformity of data, from the geometrical approximations, and from its filtering strategy. Above 50 dimensions, the sequential search becomes faster than the

VA-File. With 100 dimensions, the sequential search needs 256 seconds and the VA-file 336. These results are confirmed by those given in the article presenting the VA-File (see their Figure 12). This article says that above 45 dimensions, the approximation files becomes too large to fit in main memory, increasing the number of I/Os and the overall response time.

Regardless of the nature of the data stored in the database (real or uniform), the response times needed to search the 10 nearest neighbours of 150 descriptors in a small database are big: around a minute for both the sequential scan and the VA-File with 24 dimensions. These response times are above what one might tolerate if these techniques were part of a real system. The next experiment further investigates the influence of the size of the database on the response times.

5.5. Experiment 3: Influence of the Database Size

Figures 10 and 11 show the impact of the size of the database on the response times of the three techniques we evaluated. For this experiment, we reused the 413,412 descriptors previously computed with 24 dimensions, and generated new databases by keeping only 100,000, 200,000, 300,000 and 400,000 of them. The requests are made of the same 150 descriptors in 24 dimensions as above. We could not easily create larger databases, since the amount of real data we could use was limited. It is easy, however, to create uniform databases of arbitrary sizes. We therefore created such databases, and the larger we generated contains 1,000,000 descriptors (96 Mb), and this could correspond to more than 6500 images if we assume that an image is described by 150 local descriptors, on average.

Figure 10 shows the case with real data. What was observed in the previous experiment can again be found here. That is, the Pyramid-Tree is more expensive than other techniques, and the VA-File is slightly better than the sequential. Still, 15 seconds are needed to perform a search of a database made of only 100,000 descriptors.

Figure 11 shows the case with uniform random data. For

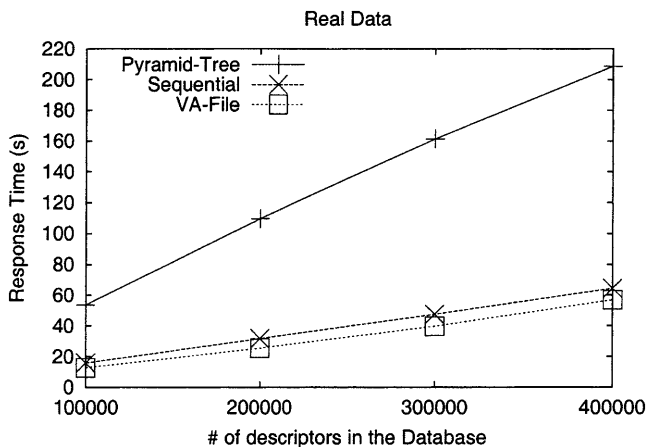


Fig. 10. 24 dimensions descriptors, 150 descriptors in each query, increasing the size of the real database.

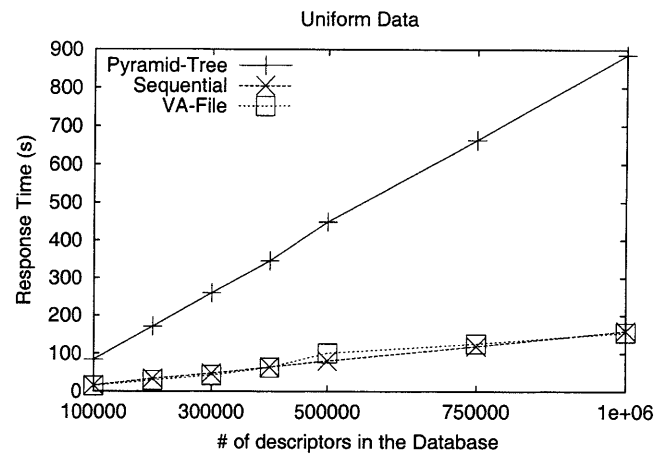


Fig. 11. 24 dimensions descriptors, 150 descriptors in each query, increasing the size of the uniform database.

a database made of 1,000,000 descriptors, the response time for the sequential scan or for the VA-File is about three minutes (160 seconds). This result clearly excludes the use of these techniques in a real system indexing millions of images (and therefore many more descriptors). Furthermore, our request has only 150 descriptors, and Section 3 states that they are, in the general case, more numerous. The next experiment focuses on this problem, and shows the influence of the number of descriptors in a request on the response times.

5.6. Experiment 4: Impact of the Number of Descriptors in a Request

All the descriptors used in this experiment have 24 dimensions. To generate the queries used here, we searched in our real database images for which 100, 200, 300 and 400 descriptors were computed. We also made up an artificial query that has only one descriptor, since this is the typical case for which the database techniques have been designed. Forging synthetic queries is trivial, and the largest one had 1000 descriptors.

The results of this experiment given in Fig. 12 (for real data) and Fig. 13 (for uniform distribution), showing again that only the VA-file and the sequential scan remain interesting. The response time, however, rapidly grows with the number of descriptors in the query. For example, 400 real descriptors cause the response time to jump to 121 seconds for the VA-File and to 185 for the sequential. With uniform data, 997 and 1042 seconds are needed, respectively, for the VA-File and the sequential with 1000 descriptors.

The number of descriptors in each query is directly related to the number of points of interest detected in the query image (see Section 3.2). This number can clearly be very large, depending on the image and on the detection strategy. It is crucial that the cost of a query having many descriptors does not increase, as illustrated by this experiment.

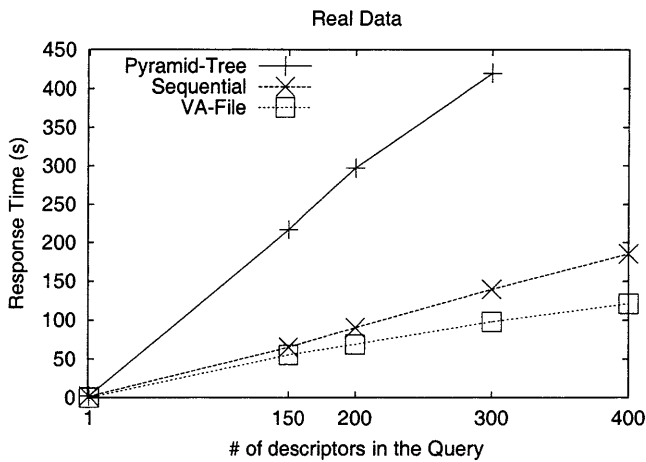


Fig. 12. Real database storing 413,412 descriptors having 24 dimensions, increasing the number of descriptors in each query.

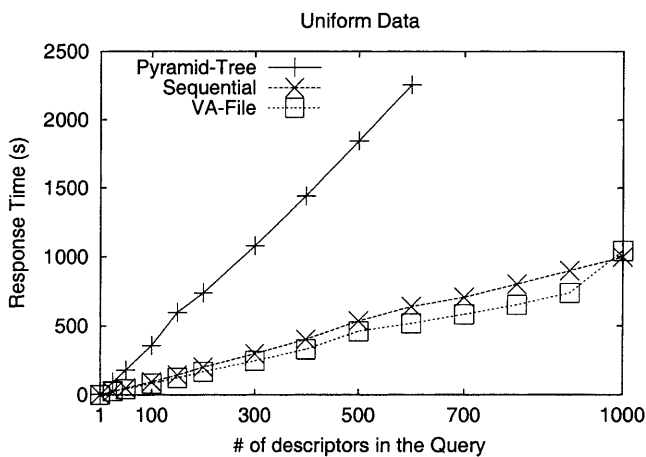


Fig. 13. Uniform database storing 413,412 descriptors having 24 dimensions, increasing the number of descriptors in each query.

6. CONCLUSION AND PERSPECTIVES

In addition to the well known slowdown caused by the dimensionality of data and by the size of the database, our performance analysis shows that additional costs severely degrade the performance of recent database indexing techniques when local descriptors are used. While this degradation is clearly seen in our experiments, worse results are expected if the techniques are to be used in a more realistic environment, where the size of the image bank is far bigger than our database (up to several Gb), where the descriptors have many more dimensions (several hundred), or where the number of descriptors used for one query is much greater (thousands).

It is therefore crucial to come up with new indexing techniques specifically designed to efficiently support the use of local descriptors. We therefore propose several research directions aimed at tempering the above mentioned effects.

Numerous Local Descriptors for a Single Query Creates Redundancy. When local descriptors are used, the image

recognition is not based on a one-to-one match between a unique descriptor stored in the database and a single descriptor used in the query. Rather, recognition is based on multiple searches, each returning information which, once accumulated and post-processed (cross-checking), gives the final answer. Some images stored in the database will belong to this final answer, because several descriptors of the query are matched with several descriptors associated with these images. There is therefore a certain form of redundancy in the information used during the complete search process (because all these query descriptors are associated with a *single* query image), and in the information returned (because an image is found similar, since many of its descriptors match). It is possible to use this redundancy in (at least) two ways.

First, the search process can be restricted so that it checks only the descriptors that are in the same cell as that in which each query descriptor falls. This avoids the typically and mandatory lookup of all neighbouring cells, which is known to be expensive, since many cells must be visited. If the search process returns, for *each* query descriptor, only the nearest neighbours that are *in the same query cell*, and ignores other potential neighbours that are in adjacent cells, then the result of each query is clearly a rough approximation of what would be returned if the normal search process was enforced. The quality of this approximation, however, is improved as time goes by, since many query descriptors are used to obtain images that are similar to *one* image. Cross-checking what is returned by each individual search is a natural way to consolidate the final answer, and fully uses the observed redundancy.

This search process therefore only needs to determine the cell in which one query descriptor falls, sequentially fetch the descriptors stored in that cell and compute the actual distances.⁴ This simple strategy is repeated until all query descriptors are used this way. It has the interesting property of trading the accuracy (of the final result) for efficiency.

The second way to use the redundancy is to stop the search before having used all of the query descriptors. In this case, the search is a greedy algorithm, and each partial result returned by each individual query is immediately processed and updates the (in progress) final result. When this current (incomplete) final result has a high probability of being the complete final answer, the search is stopped, the remaining query descriptors are skipped, and the result is returned to the user.

Note that this second search strategy is orthogonal to the first one mentioned above. Both might be combined to search only the relevant cells (ignoring adjacent cells) for a limited number of query descriptors.

Exploit the Distribution of Data to Accelerate the Queries. Not all descriptors carry the same amount of information: some are associated with many images, others are more rare. Therefore, the matching of two descriptors returns more or less discriminative information, making the

⁴ It is unlikely that *all* query descriptors fall in empty cells.

associated database image more or less likely to be part of the final result. In this case, using a Bayesian formalism may help in determining the probability for each match to help converging towards the final result. It is consequently possible to sort the descriptors in the query so that it starts with descriptors that are most informative. It is therefore possible to stop the search as soon as the probability of having the final answer is high enough, or as soon as the search starts using the descriptors that do not help in converging. This technique has the interesting property of refining the search as time goes by. In addition, the search accuracy can easily be made controllable by the user.

Change the Management of Memory to Benefit from Consecutive Queries. Traditional techniques assume that a single search within the database is sufficient to return the final answer. Therefore, what is fetched in memory during a search only benefits the next query by chance: if the second query is lucky enough to use some of the data brought in memory by the first query, than its response time is enhanced because some data is already cached. A better mechanism can be designed when local descriptors are used. In this case, we know in advance that a large number of consecutive queries will be submitted to the database. Therefore, it may be interesting to pick the next query descriptor with respect to what is already in the cache. That is, the next descriptor used to query the database can be the one which is most likely to have its nearest neighbours *already in memory*, brought in by previous descriptors. Therefore, instead of consuming all the query descriptors sequentially as the natural search process does, descriptors are picked in a memory-conscious way.

Using Several Low-Dimension Indexes Instead of a Unique High-Dimension Index. It is known that the cost of content-based retrieval grows quickly when the dimension of data increases. It is therefore potentially interesting to evaluate whether querying many low-dimension indexes in parallel instead of a unique high-dimension index gives good results. These 'small' indexes must be constructed in such a way, and their use must be such that the result they return is identical to what a regular index would return.

A small index would store the same descriptors as those stored by the large index. These, however, would be truncated, and would only keep specific dimensions chosen with care. A particular dimension might be kept by more than one small index. A query would then have to be transformed into multiple sub-queries, each interrogating a given (small) index. If these indexes are physically stored on different machines, then large grain parallelism is possible.

This scheme tries to limit the problem of the dimensionality curse by enforcing multiple interrogations of low-dimension data for which efficient indexing schemes exist. On the other hand, additional processing steps are needed, and the global size of the database (i.e. the size occupied by all the descriptors, and not by the images) is increased. These disadvantages must be placed in perspective with the potential enhancements provided by the parallelisation and the efficiency of each sub-query.

References

1. Nievergelt J, Hinterberger H, Sevcik KC. The grid file: An adaptable, symmetric multikey file structure. *ACM Trans Database Systems* 1984; 9(1):38-71
2. Robinson JT. The K-D-B-Tree: a search structure for large multidimensional dynamic indexes. *Proc ACM SIGMOD, Ann Arbor, Michigan, April 29-May 1 1981*; 10-18
3. Henrich A. The LSD^b-tree: An access structure for feature vectors. *Proc 14th ICDE, Orlando, FL, 1998*; 362-369
4. Guttman A. R-trees: A dynamic index structure for spatial searching. *Proc ACM SIGMOD, Boston, MA, 1984*; 47-57
5. Berchtold S, Keim DA, Kriegel H-P. The X-tree: An index structure for high-dimensional data. *Proc 22nd VLDB, Mumbai (Bombay), India 1996*; 28-39
6. White DA, Jain R. Similarity indexing with the SS-tree. *Proc 12th ICDE, New Orleans, LA, 1996*; 516-523
7. Katayama N, Satoh S. The SR-tree: An index structure for high-dimensional nearest neighbor queries. *Proc ACM SIGMOD, Tucson, AZ, 1997*; 369-380
8. Lin K-I, Jagadish HV, Faloutsos C. The TV-tree: an index structure for high-dimensional data. *VLDB Journal* 1994; 3(4):517-542
9. Berchtold S, Böhm C, Kriegel H-P. The Pyramid-Tree: Breaking the curse of dimensionality. *Proc ACM SIGMOD, Seattle, WA, 1998*; 142-153
10. Faloutsos C, Barber R, Flickner M, Hafner J, Niblack W, Petkovic D, Equitz W. Efficient and effective querying by image content. *J Intelligent Infor Syst.* 1994; 3:231-262.
11. Florack LMT, ter Haar Romeny B, Koenderink JJ, Viergever MA. General intensity transformation and differential invariants. *J Math Imaging and Vision* 1994; 4(2):171-187
12. Rothwell CA. *Object Recognition Through Invariant Indexing.* Oxford Science, 1995
13. Binford TO, Levitt TS. Quasi-invariants: Theory and exploitation. *Proc DARPA Image Understanding Workshop 1993*; 819-829
14. Finlayson GD, Drew MS, Funt B. Color constancy: Generalized diagonal transforms suffice. *J Opt Soc Am A* November 1994; 11(11):3011-3019
15. Slater D, Healey G. The illumination-invariant recognition of 3D objects using color invariants. *IEEE Trans Pattern Analysis and Machine Intelligence* 1996; 18(2):206-210
16. Mundy JL, Zisserman A, (eds). *Geometric Invariance in Computer Vision.* MIT Press, Cambridge, MA, 1992
17. Mundy JL, Zisserman A, Forsyth D. Application invariance in computer vision. *Lecture Notes in Computer Science* 1993; 825
18. van Gool L, Moons T, Ungureanu D. Affine/photometric invariants for planar intensity patterns. *Proc 4th Euro Conf on Computer Vision, Cambridge, UK, 1996*; 642-651.
19. Stricker M, Swain M. The capacity of color histogram indexing. *Proc Conference on Computer Vision and Pattern Recognition, Seattle, WA, 1994*
20. Huang J, Ravi Kumar S, Mitra M, Zhu WJ, Zabih R. Image indexing using color correlograms. *Proc Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA, June 1997*; 762-768
21. Schmid C, Mohr R. Local grayvalue invariants for image retrieval. *IEEE Trans Pattern Analysis and Machine Intelligence*, May 1997; 19(5):530-534
22. Schmid C, Mohr R, Bauckhage Ch. Comparing and evaluating interest points. *Proc 6th International Conference on Computer Vision, Bombay, India, January 1998*; 230-235
23. Bigün J, Granlund GH, Wiklund J. Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE Trans Pattern Analysis and Machine Intelligence* August 1991; 13(8):775-790

24. Harris C, Stephens M. A combined corner and edge detector. *Alvey Vision Conference* 1988; 147–151
25. Dufournaud Y, Schmid C, Horaud R. Matching images with different resolutions. *Proc Conference on Computer Vision and Pattern Recognition*, Hilton Head Island SC, June 2000; 1:612–618
26. Poynton CA. *Frequently asked questions about color* 1997
27. Agrawal R, Gehrke J, Gunopulos D, and Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. *Proc ACM SIGMOD*, Seattle, WA, 1998; 94–105
28. Hinneburg A, Keim DA. Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering. *Proc 25th VLDB*, Edinburgh, Scotland, 1999; 506–517
29. Weber R, Schek H-J, Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. *Proc 24th VLDB*, New York, 1998; 194–205
30. Mohr R, Gros P, Schmid C. Efficient matching with invariant local descriptors. *Proc Joint IAPR International Workshops SSPR98 and SPR98: Advances in Pattern Recognition*, Sydney, Australia: *Lecture Notes in Computer Science*, Springer-Verlag, August 1998; 1451:54–71

Laurent Amsaleg received his PhD in June 1995 from the University of Paris 6, France. He worked at INRIA from 1989 to October 1995 in the Rodin project, where his research interests included relational and object-oriented databases, garbage collection, object clustering, operating systems, micro-kernels and single-level stores. He then spent 18 months in the Database group of the University of Maryland (MD, USA), designing flexible database query execution strategies. He subsequently joined the IRISA lab in Rennes (France), and works in the TEMICS Group. His research interests include the database and system sides of content-based retrieval systems, e.g. multi-dimensional indexing, prefetching, caching, and other low-level functions.

Patrick Gros received an engineering degree from the École Polytechnique in 1988 and from École Nationale Supérieure de Techniques Avancées (France) in 1990. He received his PhD from the University of Grenoble, France in 1993. After six years in the GRAVIR lab at Grenoble and one year at the Robotics Institute of CMU (Pittsburgh, PA), in 2000 he moved to the IRISA lab of Rennes, France, where he works in the VISTA research group. His research interests concern various geometric and photometric techniques applied to image matching and indexing. He is involved in several projects concerning image and video archiving and object recognition.

Correspondence and offprint requests to: L. Amsaleg, IRISA–CNRS, Campus de Beaulieu, 35042 Rennes cedex, France. Email: Laurent.Amsaleg@irisa.fr