Optimizing Progressive Query-By-Example over Pre-Clustered Large Image Databases

Anicet Kouomou-Choupo
IRISA - Campus de Beaulieu
35042 Rennes cedex, France
+33 299847157
Anicet.Kouomou_Choupo@irisa.fr

Laure Berti-Équille
IRISA - Campus de Beaulieu
35042 Rennes cedex, France
+33 299847390
Laure.Berti-Equille@irisa.fr

Annie Morin
IRISA - Campus de Beaulieu
35042 Rennes cedex, France
+33 299847222
Annie.Morin@irisa.fr

ABSTRACT

The typical mode for querying in an image content-based information system is query-by-example, which allows the user to provide an image as a query and to search for similar images (*i.e.*, the nearest neighbors) based on one or a combination of low-level multidimensional features of the query example. Off-line, this requires the time-consuming pre-computing of the whole set of visual descriptors over the image database. On-line, one major drawback is that multidimensional sequential NN-search is usually exhaustive over the whole image set face to the user who has a very limited patience. In this paper, we propose a technique for improving the performance of image query-by-example execution strategies over multiple visual features. This includes first, the pre-clustering of the large image database and then, the scheduling of the processing of the feature clusters before providing progressively the query results (*i.e.*, intermediate results are sent continuously before the end of the exhaustive scan over the whole database). A cluster eligibility criterion and two filtering rules are proposed to select the most relevant clusters to a query-by-example. Experiments over more than 110 000 images and five MPEG-7 global features show that our approach significantly reduces the query time in two experimental cases: the query time is divided by 4.8 for 100 clusters per descriptor type and by 7 for 200 clusters per descriptor type compared to a "blind" sequential NN-search with keeping the same final query result. This constitutes a promising perspective for optimizing image query-by-example execution.

1. INTRODUCTION

The management of large collections of images adds the problem of efficiently handling and searching an overwhelming volume of digital contents and their associated descriptions. Challenges include balancing between rich content description and efficient storage with fast access methods at various levels (from pixels to semantics). Query-by-example exploits and combines visual descriptors whose computing costs increase proportionally to the database size. The exhaustive processing of a very great number of images with NN-searches turns to be quickly unaffordable to users desiring quick response time. Researchers from computer vision community have proposed a plethora of image description and retrieval methods that represent each image as a set of low-level features and provide users with a query-by-example similarity search to explore the database contents. However, the convergence between computer vision and database communities is not yet achieved regarding the two following points:

1. The main current research work trend deals with algorithmic approaches for nearest neighbors searches and early stopping approaches that focus on multidimensional but mainly mono-descriptor datasets [3],[14],[23].
2. Contrary to traditional databases where many works have been carried out on query optimization and more recently on adaptive query processing [1],[9], very few works have been proposed on image query-by-example optimization.

Motivated by such observations, we present in this paper a case study of how to apply a database optimization approach to an image query-by-example problem under the multi-descriptor paradigm. More recently, Kiranyaz and Gabbouj [11] defined the notion of progressive query-by-example allowing intermediate results before the end of the whole database processing but the approach was limited to the periodic execution of the same sub-query with several descriptors on different portions of multimedia databases (images and videos) with no emphasis on scheduling the data subsets to process by NN-searches, or on merging the results from heterogeneous descriptors. Our research builds upon the implementation of progressive content-based image retrieval and upon previous experiments [13] that extended the work of Kiranyaz and Gabbouj. For querying very large image databases, our assumption is that *i)* pre-clustering the image descriptions, *ii)* scheduling the NN-searches over the ranked set of relevant descriptors, and *iii)* presenting continuously the intermediate results during the running query should be more efficient and should offer more flexibility to users. We propose then a system of image retrieval that organizes image features into clusters and schedules their processing for progressive image retrieval. The system combines multiple visual global features and uses filtering rules to select the most relevant clusters to scan in priority.

The rest of the paper is organized as follows: Section 2 presents a synthesis of related work on indexing techniques, selection and scheduling clusters for image content-based retrieval. Our system is described in Section 3. Experiments and results are reported in Section 4. We conclude the paper in Section 5 and also present our research perspectives.

2. RELATED WORK

In computer vision community, many methods have been proposed to extract low-level visual images features. Some works deal with local descriptor extraction and try to build features that are invariant to geometric transformations and robust to illumination changes [18],[22]. Others suggest global descriptors that are more simple information to cope with and less memory space and time consuming [16],[21]. However, one of the basic questions to address in an image database system is to define when two images may be considered similar. Similarity of images has been characterized in the literature through three important features: colour, texture and shape [20] and systems that combine computer vision techniques to measure the similarity between images are also proposed [25]. In this paper, we are essentially dealing with query-by-example on general real-world images. Global features remain best suited in this case and we privilege merging results of several descriptors instead of defining new distances on a combination of features. We assume that each feature is effective separately and that merging results is appropriate to answer multi-descriptor queries. We then focus in this paper on the efficiency of our approach.

The second question to address is indexing high-dimensional descriptors. Large image databases have to be indexed in order to make the retrieval efficient. The general principle of indexation schemes is to gather in the same cell all feature vectors that are collocated in the multidimensional space. Cells can then be included within a specific indexing structure like trees. Indexation techniques that are either based on data partitioning (such as R-Tree [10]) or on space partitioning (such as k-d-Tree [2] or GridFile [15]) are well adapted to low-dimensional vectors (≤ 16 dimensions according to [24]). High dimensional data spaces have some specific properties which severely affect retrieval methods, making a sequential search perform better. These property have been studied and reported in [4],[24]. New indexation schemes have to be defined and adapted to the high-dimensional vectors. Cornacchia et al [6] propose a system based on *Relational Array Mapping* that naturally extends existing database functionalities in terms of indexing to cope with multidimensional arrays. The strength of the system lies on a specific algebra (Relational Array Mapping Algebra) to generate optimized query execution plans and to improve expressiveness of the database system. In our work however, we stay in multidimensional space: we organize images into clusters and we determine a ranked list of clusters per query for efficiently and effectively retrieving the most similar images according to all the available descriptors in the database. This ranked list of the most eligible clusters defines a query execution strategy for the given query-by-example.

Once the questions of similarity of images and indexing high-dimensional features have been addressed, the objective is to define and to elaborate efficient strategies for image retrieval. Retrieval methods are closely related to the underlying indexation schema. Nearest Neighbors algorithms typically use the geometrical properties of cells to eliminate those cells which can not have any impact on the result of the current query [5]. Filtering rules are sometimes used for searching exact nearest neighbors and for eliminating irrelevant cells, avoiding the subsequent analysis of all the vectors they contain, which, in turn, reduces response time. But the “curse of dimensionality” phenomenon makes these filtering rules ineffective on high-dimensional datasets [12],[24],[4],[5]. There is therefore an increasing interest in performing approximate NN-searches, where the quality of result is traded off against reduced query execution time. Weber and Böhm with VA-File [23] and Li *et al.* with Clindex [14] perform approximate NN-searches by interrupting the search after having accessed an arbitrarily fixed number of cells. These two techniques are efficient in terms of response time, but give no clue on the quality of the result returned to the user. Berrani *et al.* [3] control the precision of the search by setting at run time the maximum probability for a vector that would be in the exact answer set to be missing in the approximate set of answers. They use approximated cells' radii which improve the effectiveness of filtering rules and speed up the NN-search. Our work does not currently use approximate cells' radius for mono-descriptor NN-search. Our focus is at a higher multi-descriptor level over a pre-clustered large database. Our approach is nevertheless complementary. We defined approximate filtering rules over eligible clusters with all descriptors taken together and experimentally quantify the introduced approximation on the result quality.

3. PROGRESSIVE CONTENT-BASED IMAGE RETRIEVAL

Image query-by-example may use a combination of several types of visual descriptors whose time-consuming computing costs increase proportionally to the database size. At the query-time, the exhaustive processing of a very great number of images with NN-search turns to be quickly unaffordable to users desiring quick response time. In this context, our twofold assumption is: *i)* pre-clustering the image features of large databases and scheduling the NN-search over a ranked set of relevant clusters of descriptors should improve the performance of image query-by-example and *ii)* sending continuously the intermediate results of the search in progress should offer more flexibility to the users.

3.1 Image Database Pre-Clustering

Image files are gathered on the hard disk and global descriptions of the images are pre-computed for m types of descriptors d_i ($d_i \in D$, $i=1,2,\ldots,m$). For each descriptors type d_i, the image descriptions are organized into clusters with a *k-means*-like clustering method which reuse the *k-means* principle but replace the computed virtual centroid by the nearest existing image (which will be called centroid for the sake of simplicity). This approximation is introduced to allow an appropriate computing of distances between image query and clusters' centroids for a given descriptor type. Several clusters characteristics are stored in a binary file: their size in term of number of elements they contains, their centroid, and their position in the binary file. For each descriptor type, an image is described by a vector with variable length per descriptor type. On the one hand, the choice of the number of clusters as input for the *k-means* algorithm [8] remains a well-known difficult problem. Our first experimental approach was to take a number of clusters per descriptor type in order to have clusters with almost the same size. Our previous experiments reported in [13] show that the equal number of clusters per descriptor type does not change the results from one descriptor type to another. Section 4 will give details on this choice. On the other hand, *k-means* is not effective dealing with outliers, but our interest is focused on search strategies and outliers' management is not in the scope of this paper. Moreover, it has been shown that minimizing the search time suggests to generate uniformly sized clusters [19].

3.2 Image Querying by Example

Based on the pre-clustering of the image database, our image retrieval process includes the following steps (Figure 1): (i) the user submits an image query to the system and all the features available for image description are computed from the image query; (ii) clusters characteristics related to the query are scanned. For example, distances from the centroid of clusters to the image query, maximum and minimum distances from the query to clusters are computed at this step; (iii) the clusters are ranked based on their characteristics; (iv) filtering rules are applied to select the most relevant clusters and to speed up the retrieval process; (v) clusters scanning is scheduled to provide to users the best results as soon as possible: progressive query is executed based on the scheduling of the clusters scan and intermediate results are continuously presented to the user ; the process loops from the step (iv) until there does not remain any cluster to analyze.

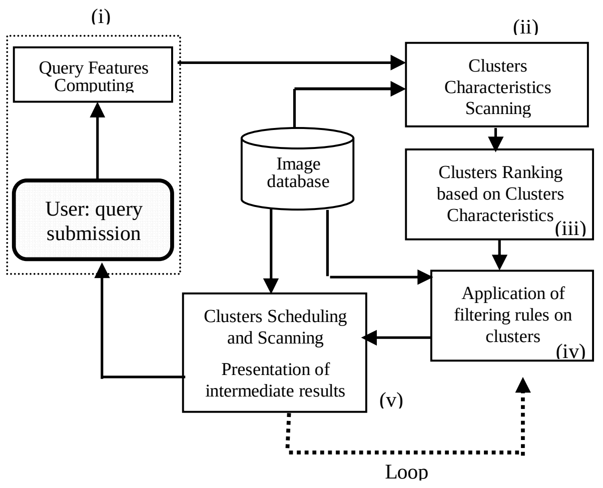

(i)
Query Features Computing
User: query submission
(ii)
Clusters Characteristics Scanning
Image database
Clusters Ranking based on Clusters Characteristics
(iii)
Application of filtering rules on clusters
(iv)
Clusters Scheduling and Scanning
Presentation of intermediate results
(v)
Loop

Figure 1. Steps in the progressive retrieval process.

In a query-by-example scenario with a similarity search target (comparable to searching similar images on the web), users submit an image as a query to the system to get in return the most similar images to the image-query according to particular visual criteria. Users may know how to choose visual descriptors. However, it generally happens that they do not have any idea of the most discriminating descriptor to use for their query. For such a case, our approach finds and assigns priorities on the clusters of image descriptions and, consequently, proposes continuously a list of ranked clusters for optimizing the query-by-example execution. The following section formalizes the key notions of our approach that are: the cluster eligibility criterion and the filtering rules upon which our method is based for significantly improving the performance of image query-by-example.

3.3 Selecting and Scheduling Feature Clusters

Let *m* be the number of features and *n* the number of clusters for each descriptor type. We choose to compute, for each cluster C_{ij} *(j=1, 2,..., n)* of descriptor type d_i *(i=1,2,…,m)*, the distance from its centroid to the submitted image-query, noted I_q. This distance is noted for simplification: $Dist(I_q,C_{ij})$. Computing the distance depends on the nature of the multidimensional feature. In the case of MPEG-7 global descriptors, the distance is generally a quadratic or a weighted Manhattan distance. We define the cost of a given cluster C_{ij}, to be the vector dimension of the descriptor type d_i multiplied by the number of vectors that the cluster C_{ij} contains. This cost is noted $Cost(C_{ij})$. We normalize $Dist(I_q, C_{ij})$ and $Cost(C_{ij})$ for all d_i *(i=1,2,…,m)* in order to make them independently comparable. $NormDist(I_q,C_{ij})$ and $NormCost(C_{ij})$ indicate respectively the normalized distance and the normalized cost of the cluster C_{ij}. These measures are defined by Eq. (1):

$$NormX = \frac{X - E(X)}{\sigma(X)} \ with \ \begin{cases} X = Dist(I_q, C_{ij}) \ or \\ X = Cost(C_{ij}) \end{cases} \tag{1}$$

$E(X)$ and $\sigma(X)$ are respectively estimated by mean and standard deviation of *X* for the descriptor type d_i. These normalized measures are combined to define the eligibility criterion of each cluster C_{ij} for the image-query I_q such as:

$$\begin{aligned} eligibility(I_q, C_{ij}) = \alpha & \left| NormDist(I_q, C_{ij}) - \min_{\substack{i=1..m \\ j=1..n}} NormDist(I_q, C_{ij}) \right| \\ & + \beta \left| NormCost(C_{ij}) - \min_{\substack{i=1..m \\ j=1..n}} NormCost(C_{ij}) \right| \\ & with\ \alpha, \beta \in \{0,1\}\ and\ \alpha + \beta \neq 0 \end{aligned} \tag{2}$$

This formula is limited to the extreme cases of behavior of normalized measures studied separately, and the equally weighted combination of theses measures. The study can be easily extended to the general case of a weighted combination.

We also propose two filtering rules in our retrieval process. More formally, for a descriptor type d_i, let R_{ij} be the radius of the cluster C_{ij}. It is equal to the distance between the centroid of C_{ij} and its farthest point. We define the minimum distance $d_{\min}$ between the image query I_q and the cluster C_{ij} by Eq. (3):

$$d_{\min}(I_q, C_{ij}) = \begin{cases} Dist(I_q, C_{ij}) - R_{ij} & if\ Dist(I_q, C_{ij}) > R_{ij} \\ 0 & else \end{cases} \tag{3}$$

The first rule (R1) has been adapted from [3]. It consists of discarding all clusters which the minimum distance to the image query I_q is greater than a computed threshold. It is stated as follows:

$$\begin{aligned} & \text{R1} : \mathit{If} \\ & d_{\min}(I_q, C_{ij}) \geq \min\Big(\min_{\substack{p=1..n \\ Card(C_{ip}) \geq k}} (Dist(I_q, C_{ip}) + R_{ip}), d(I_q, kNN) \Big) \\ & \qquad \mathit{then}\ C_{ij}\ \mathit{is\ irrelevant} \end{aligned} \tag{4}$$

where $Card(C_{ip})$ is the number of vectors of the cluster C_{ip} *(p=1,2,...,n)*, *k* the number of nearest neighbors to search, and $d(I_q, kNN)$ the distance between the image query I_q and the current k^{th} nearest neighbour. The rule (R1) simply exploits a shape property of clusters, ensuring the correctness of retrieved results, comparing to the sequential search. It is however possible to use a rule stronger than the rule (R1) which exploits the sensibility of the chosen clustering method regarding outliers in order to discard from the search, all clusters which the minimum distance to an image query is not null. This second rule is stated as follows:

$$\text{R2: } \textit{If } d_{\min}(I_q, C_{ij}) > 0 \textit{ then } C_{ij} \textit{ is irrelevant} \tag{5}$$

If the query belongs to the image database, it then belongs to one cluster for a given descriptor and intuitively, it is very probable the cluster contains some neighbors of the query. According to the definition of $d_{\min}$ (see Eq. (3)), the rule (R2) then indicates the condition under which a cluster may be supposed far from the query, and thus irrelevant.

3.4 Merging of Intermediate Query-by-example Results

Our content-based image retrieval system searches for the most similar images and gradually uses the available descriptors. Intermediate results are produced as soon as the NN-search has been done over one of the most eligible clusters and they are progressively merged together with the intermediate results of the previous searching phases. The fusion of the result lists obtained from the similarity search on various multidimensional descriptors is a complex problem. We use the merging function introduced in our previous work [13].

Consider at a given searching phase *p*, the similarity search is done over the cluster $C_{i_0 j}$ of the descriptor type d_{i_0}. We note $\{d_1, d_2, \ldots, d_{m_1}\}$ the subset of m_1 descriptors type corresponding to the clusters that have already been processed at the end of the searching phase *p* ($m_1 \leq m$). The *Top N* list of intermediate results on the descriptor type d_i is noted l_i, $1 \leq i \leq m_1$. For the searching phase *p,* the list l_{i_0} is updated by taking into account the processing of cluster $C_{i_0 j}$. Then, the list l_{i_0} is merged with the other lists l_i ($1 \leq i \leq m_1$, $i \neq i_0$) in order to obtain the final *Top N* list of results. To cope with the problem of *Top N* lists fusion, we define a scoring function for an image *I* such as:

$$S_f(I) = \frac{1}{2}\left(\sum_{i=1}^{m_1} \alpha_i \left(f_{d_i}(I) + S_{d_i}(I)\right)\right)$$
$$\textit{with } S_{d_i}(I) \in [0,1], \quad f_{d_i}(I) = \begin{cases} 1 & \textit{if } \; I \in l_i \\ 0 & \text{if } \; \mathrm{I} \notin \mathrm{l_i} \end{cases} \quad \textit{and } \sum_{i=1}^{m_1} \alpha_i = 1 \tag{6}$$

$S_{d_i}(I)$ is the score of the image *I* for the descriptor type d_i. It is proportional to the similarity between the query-by-example I_q and *I* according to the descriptor type d_i. α_i *(i=1..m₁)* are weights of descriptors type d_i.

This function takes into account the presence of an image in the list of results for a given descriptor type and the score of the image in that list. Hence, all images appearing more often and that having high scores in individual lists are relevant. The main advantage of the proposed fusion function is its granularity that allows merging intermediate result lists after each cluster scan whatever the descriptor type may be. Nevertheless, our experimental study on the efficiency is comparative: we use the same fusion function for both approaches, for the “blind” sequential search and for our progressive and selective execution strategy. No bias is introduced regarding to the fusion of results lists. Moreover, other methods of result aggregation as OMEDRANK [7] can be adapted and used.

4. EXPERIMENTS

To validate our propositions, we set up experiments with 110,291 still images and five global MPEG-7 descriptors of color (named *ColorLayout, ScalableColor*), texture (*HomogeneousTexture, EdgeHistogram*) and shape (*RegionShape*). In the following, descriptor types are abbreviated respectively as *CLD* and *SCD* for color, *HTD* and *EHD* for texture, and *RSD* for shape. CLD descriptor type uses L_2 distance but others are defined with L_1 distance. The images are gathered into clusters for each descriptor type with the *k-means*-like algorithm described in Section 3.1. The images of our database are provided by a photo agency. We do not have any detail on their source to consider a robust plan of sampling. The images are heterogeneous and mostly dedicated to press agencies and professional users. One hundred of images that do not belong to the database have been chosen randomly as queries examples for testing and evaluating our method implemented in C++ under Linux (Pentium 4, 3 GHz, with 1 Gb of main memory). The objective is to organize the image set for optimizing the image query-by-example execution. For each image query, we search for their 15 nearest neighbors.

Without user's relevance feedback, it is meaningless to evaluate precision and recall and, also the subjective quality of the retrieved results with no prior knowledge on the application context or on the underlying search targets. For this reason, we mainly focused, in these experiments, on the problem of performance, and we proposed to measure quality relatively to the final result list obtained at the end of a "blind" sequential NN-search. This definition of the quality is of course relative but gives a good idea of the gain on query time of our approach without quality loss compared to the classical sequential search. The relative quality is computed by counting the number of common images in each intermediate result list and those in the final result obtained at the end of the "blind" sequential search.

In the following section 4.1, we test our method by first studying parameters α and β of the cluster eligibility formula in Eq. (2). Then, in the section 4.2, we evaluate the two filtering rules proposed in (4) and (5). A short study is made in section 4.3 to show the interest of combination of feature for content-based image retrieval.

4.1 Cluster Eligibility

We performed experiments on the eligibility measure for different values of the necessary number of clusters given as input to the *k-means*-like algorithm. For the sake of simplicity, we choose to detail the case of 100 clusters per descriptor type. Nevertheless, we made experiments on 10, 50 and 100 clusters per descriptor type that showed that the behavior we will describe further is similar over the database whatever the number of clusters may be. Figure 2 and Figure 3 represent respectively the variations of the relative quality (compared to a "blind" sequential NN-search) and the corresponding response time according to the searching phases. A searching phase corresponds to the end of a cluster scan with NN-search. Values in parenthesis correspond to different values for the parameters α and β in Eq. (2): (1,0) is associated to α=1 and β=0; (1,1) corresponds to α=1 and β=1; and (0,1) to α=0 and β=1.

The eligibility based on the distance from the cluster centroid to the image-query obviously improves the speed of the global search processing: the same result than the one obtained in a "blind" sequential search is reached much more quickly (as shown in Figure 2): a "blind" sequential search will be carried out over 500 clusters (100 clusters × 5 descriptors) before sending the final *Top N* list of results. In our approach, this final list of results can be reached after the scan of 110 clusters (see the grey circles and the intersecting dotted lines on the graphs).

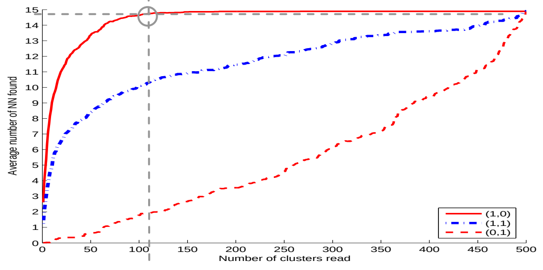


Figure 2. Relative quality of results depending on the eligibility of clusters: 100 clusters per descriptor type.

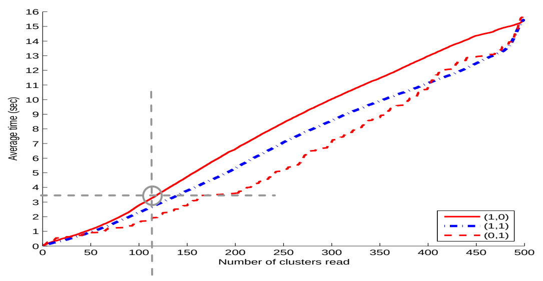
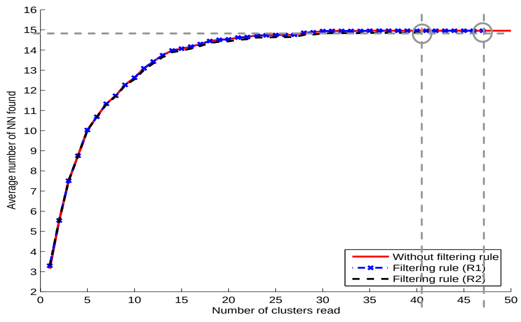


Figure 3. Query by example execution time depending on the eligibility of clusters: 100 clusters per descriptor type.

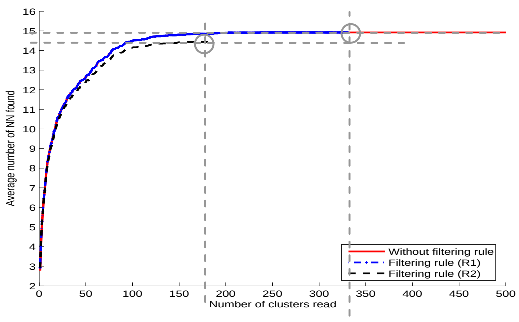
For the case of 100 clusters per descriptor type, this corresponds to a gain of about 77% on the average time (*i.e.*, 3.5s instead of 15s) compared to a "blind" NN-search as it is shown in Figure 3.

4.2 Effectiveness of Filtering Rules

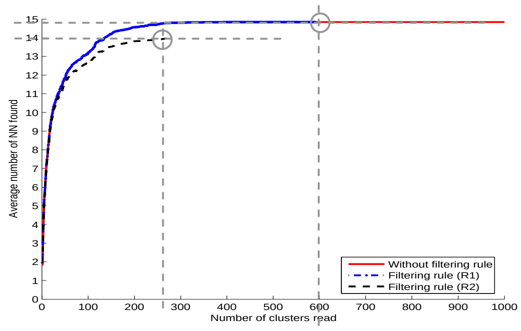
The effectiveness of the filtering rules defined previously by (4) and (5) has been compared using three values of the number of clusters per descriptor type, respectively 10, 100, and 200 clusters per descriptor type. We have then plotted variations of the relative quality and the corresponding cumulated time per searching phase in these three cases (Figure 4(a-c) and Figure 5(a-c)). Figure 4 confirms that the filtering rule (R1) assures the correctness of the retrieved results. Our experiments show that the approximation introduced by the filtering rule (R2) tends to be greater due the reduction of clusters radius as the number of clusters per descriptor increases. Concerning the effect of using the filtering rules on the searching time, our experiments show that the searching time decreases as the number of clusters per descriptor increases. The filtering rules (R1) and (R2) are then more efficient with small clusters radius.



(a) 10 clusters per descriptor type



(b) 100 clusters per descriptor type



(c) 200 clusters per descriptor type

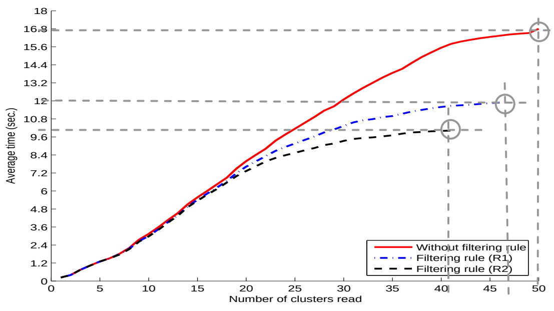
Figure 4. Comparison of the filtering rules on the average number of NN found.

For the case of 10 clusters per descriptor type, Figures 4(a) and 5(a) show that: *i)* using a "blind" sequential search, the system retrieves 15 nearest neighbors over 15 in 16.8 seconds as average time; *ii)* using the rule (R1), the system retrieves 15 nearest neighbors over 15 in 12 seconds, which represents the sequential searching time divided by 1.4; *iii)* using the rule (R2), the system retrieves 15 nearest neighbors over 15 in 10 seconds, which represents the sequential searching time divided by 1.68.

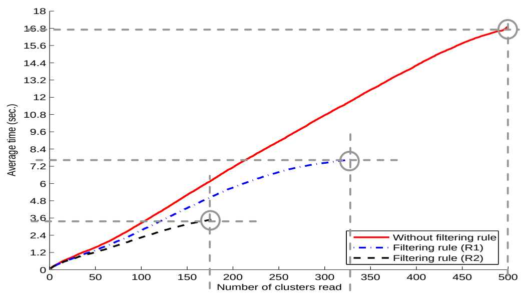
For the case of 100 clusters per descriptor type, Figures 4(b) and 5(b) show that: *i)* using a "blind" sequential search, the system retrieves 15 nearest neighbors over 15 in 16.8 seconds as average time; *ii)* using the rule (R1), the system retrieves 15 nearest neighbors over 15 in 8 seconds, which represents the sequential searching time divided by 2.1; *iii)* using the rule (R2), the system retrieves 14.5 nearest neighbors over 15 in 3.5 seconds, which represents the sequential searching time divided by 4.8.

For the case of 200 clusters per descriptor type, Figures 4(c) and 5(c) show that: *i)* using a "blind" sequential search, the system retrieves 15 nearest neighbors over 15 in 16.8 seconds as average time; *ii)* using the rule (R1), the system retrieves 15 nearest neighbors over 15 in 6.1 seconds, which represents the sequential searching time divided by 2.76; *iii)* using the rule (R2), the system retrieves 14 nearest neighbors over 15 in 2.4 seconds, which represents the sequential searching time divided by seven.

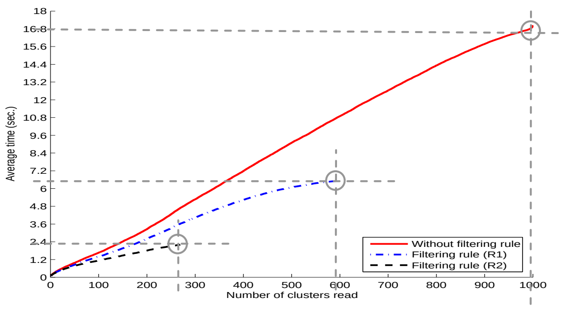
We then conclude that filtering rule (R2) is more efficient than filtering rule (R1) in terms of searching time according to Figure 5. Filtering rules (R1) and (R2) improve the efficiency of retrieval. While the rule (R1) preserves the correctness of results, the important gain of time obtained with using the rule (R2) is compensated by a light loss of result quality. It is interesting to evaluate the impact of the rules on the clusters scanning for each descriptor type.



(a) 10 clusters per descriptor type



(b) 100 clusters per descriptor type



(c) 200 clusters per descriptor type

Figure 5. Comparison of the filtering rules on average time.

Tables 1 and 2 present the mean of scanned clusters at the end of NN-search for each descriptor type. Each row corresponds to one of the three cases of the given number of clusters per descriptor type (respectively 10, 100, and 200 per descriptor type). As one could expect, much less clusters are scanned using rule (R2) for the same result list.

We also realize that the rule (R1) is particularly inefficient on clusters of *EHD* and *RSD* showing in this way the sensibility of the rule to the descriptor upon which it is applied.

Table 1. Number of scanned clusters for the filtering rule (R1)

Descriptor / K Clusters	*CLD*	*SCD*	*HTD*	*EHD*	*RSD*
10×5	10	10	7	10	10
100×5	53	62	44	78	94
200×5	83	103	72	144	190

Table 2. Number of scanned clusters for the filtering rule (R2)

Descriptor / K Clusters	*CLD*	*SCD*	*HTD*	*EHD*	*RSD*
10×5	8	9	6	8	10
100×5	21	26	23	35	78
200×5	23	32	27	46	141

The number of scanned clusters for the rule (R2) gives an idea of clusters overlapping. According to Table 2, clusters of descriptors *EHD* and *RSD* overlap more than others, which explain the inefficiency of the rule (R1). The rule (R2) may be used at the beginning of the search process providing the user with a first result near to the final one and then, the rule (R1) can be used to keep on the search.

4.3 Interest of the Combination of Global Features

This subsection aims at evaluating how the combination of descriptors type may be useful in the content-based image retrieval context. Images used to perform experimentations on the global descriptors type combination interest are provided by the database MOVI[1] which contains 32 sequences. Each sequence represents the same scene (2 or 3D) taken in different conditions (variation of the intensity of the main light source, rotation, occlusion …). MPEG-7 descriptors of images of the database MOVI are computed and added to that of 110,291 images used to evaluate the performance of our retrieval system. We arbitrarily choose one image in each MOVI's sequence as query and retrieved results for each descriptor type are merged using function defined by equation (6). We then compute the traditional recall, precision and F-measure [17] of the merged results. No standard method exists to choose the appropriate number of nearest neighbors to retrieve. As sequences are of different lengths (in term of the number of images they contain), we have chosen to retrieve for each query the number of nearest neighbors equals to the length of the sequence the query belongs to. In these conditions, recall, precision, and F-measure are all equal. In the following, we will only present results based on the F-measure.

According to table 3, SCD and EHD descriptors type have the best recognition rate. Hence, color and texture descriptors perform well on our data.

Table 3. F-measure of the system with one descriptor type

Descriptor type	F-measure
CLD	0.40
SCD	0.60
HTD	0.23
EHD	0.41

[1]Available at: http://www.irisa.fr/texmex/base_images/index.html

RSD	0.30

To each descriptor type, we affected a weight proportional to its F-measure (see Table 3) and we computed F-measure of systems obtained by the combination of descriptor type. Results are consigned in Table 4. This table shows that combining 4 descriptors type like CLD-SCD-HTD-EHD or 5 descriptors type like CLD-SCD-HTD-EHD-RSD, one obtains a recognition rate of 63 %, which is better than using only SCD descriptor (60 % of recognition rate). We also remark that using combinations such that CLD-SCD or HTD-EHD does not yield better results than using SCD or EHD alone. Using HTD-EHD-RSD is not better than using HTD-EHD. We then conclude that in the CLD-SCD-HTD-EHD-RSD combination, descriptors SCD and EHD play the most interesting role in term of recognition. This conclusion confirms observations made with the table 3.

Table 4. F-measure of the system with combination of descriptors type.

Combination of descriptor type	F-measure
CLD-SCD	0.60
HTD-EHD	0.41
CLD-SCD-RSD	0.60
HTD-EHD-RSD	0.41
CLD-SCD-HTD-EHD	0.63
CLD-SCD-HTD-EHD-RSD	0.63

5. CONCLUSION

In this paper, we have presented a novel approach for improving the query-by-example execution performance over a large image database compared to a “blind” sequential NN-search using several low-level global features. The proposed approach is based on the pre-clustering of the database and on the use of the cluster eligibility measure and two filtering rules we defined to improve image query-by-example performances compared to the “blind” search while keeping the same quality of results.

Experiments over 110,291 images and five global MPEG-7 descriptors have shown that our approach provides the following interesting insights: *(i)* the progressive query-by-example processing combines advantageously several descriptors without any knowledge on the descriptor types from users; *(ii)* relevant clusters are automatically selected, ranked and scheduled for NN-search using the cluster eligibility measure; *(iii)* the use of filtering rules over multi-descriptor clusters improves the query processing: for 100 clusters per descriptor type, by dividing by 4.8 the average time of a “blind” sequential NN-search and for 200 clusters per descriptor type, by dividing by 7 the average time, which are very promising results.

The originality of the proposed approach is that it can be applied to all kind of image databases with taking advantage of the combination of several low-level global features for query-by-example execution strategies. However, several interesting issues remain open. Our perspectives in convergence of image processing and database techniques are twofold: *i)* concerning the optimization of image query-by-example execution strategies over several descriptors without *a priori* knowledge on them, we are currently exploring the techniques of adaptive query processing and working on a query-by-example cost model for improving the query time. We will pursue our experimentations on data mining techniques introduced in [13] to improve query performance. In addition, more formal study should be done to control approximations introduced by filtering rules we proposed; *ii)* concerning the evaluation of quality of results (and for extending the notion of relative quality we proposed by comparison to a “blind” NN-search) without knowledge of user’s search target, we are planning to extend our analysis and experiments to more general query scenarios with including precision, recall and users’ relevance feedback tracking to have a more interactive system. This implies that query execution has to be dynamically adapted using optimal strategies of selecting and scheduling clusters.

6. REFERENCES

[1] Babu S., Bizarro P., Adaptive Query Processing in the Looking Glass. *Proc. of the 2nd Biennial Conf. on Innovative Data Systems Research (CIDR)*, Jan. 2005.

[2] Bentley J. L., Multidimensional binary search in database applications. *IEEE Trans. on Soft. Eng.,4(5)*, 1979, 333-340.

[3] Berrani S.A., Amsaleg L., Gros P., *Approximate k-Nearest Neighbor Searches: A New Algorithm with Probabilistic Control of the Precision*. Tech. Rep. INRIA, n° 4675, 2002.

[4] Beyer K., Goldstein J., Ramakrishnan R., Shaft U., When Is "Nearest Neighbor" Meaningful? *Proc. of the 8th Intl. Conf. ICDT*, London, U. K., 1999.

[5] Böhm C., Berchtold S., Keim D., Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. *ACM Comp. Surveys*, (33)3, 2001.

[6] Cornacchia R., Ballegooij A., de Vries A. P., A Case Study on Array Query Optimization. *Proc. of the 1st ACM Workshop Computer Vision meets Databases (CVDB)*, 2004.

[7] Fagin R., Kumar R., Sivakumar D., Efficient similarity search and classification via rank aggregation, *Proc. of the Conf. ACM SIGMOD*, 2003, 301-312.

[8] Jain A. K., Dubes R. C., *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

[9] Gounaris A., Paton N., Fernandes A., Sakellariou R., Adaptive query processing. *Proc. of the Conf. BNCOD, LNCS*, vol. 2405, 2002, 11-25.

[10] Guttman A., R-trees: A dynamic Index Structure for Spatial Searching. *Proc. of the Conf. ACM SIGMOD*, 1984, 47-57.

[11] Kiranyaz S., Gabbouj M., A novel multimedia retrieval technique: progressive query (why wait?). *Proc. of Intl. Workshop of Image Analysis*, Portugal, 2004.

[12] Korn F., Pagel B., Faloutsos C., On the 'Dimensionality Curse' and the 'Self-Similarity Blessing '. *IEEE Trans. on Knowledge and Data Engineering* 13(1), 2001, 96-111.

[13] Kouomou-Choupo A., Berti-Équille L., Visual Feature Mining for Adapting Query-by-Example over Large Image Databases. *Proc. of Intl. Workshop on Multidisciplinary, Video, and Audio retrieval and Mining*, Canada, 2004.

[14] Li C., Chang E., Garcia-Molina H., Wiederhold G., Clustering for Approximate Similarity Search in High-Dimensional Spaces. *IEEE Trans. on Knowledge and Data Engineering*, 14(4), 2002, 792-808.

[15] Nievergelt J., Hinterberger H., Sevcik K. C., The GridFile: An adaptable, symmetric multikey file structure. *ACM Trans. on Database Systems, 9(1)*, 1984, 38-71.

[16] Obeid M., Jedynak B., Daoudi M., Image indexing and retrieval using intermediate features. *Proc. of the 9th ACM/ICM*, 2001, 531–533.

[17] Rijsbergen C. V., *Information Retrieval*, Butterworth, 1979.

[18] Schmid C., Mohr R., Local Grayvalue Invariants for Image Retrieval, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5), 1997, 530-534.

[19] Sigurðardottir R., Hauksson H., Þór-Jónsson B., Amsaleg L., A Case Study of the Quality vs. Time Trade-off for Approximate Image Descriptor Search. *Proc. of the 1st IEEE Intl. Workshop on Managing Data for Emerging Multimedia Applications (EMMA)*, Tokyo, Japan, 2005.

[20] Smeulders A., Worring M., Santini S., Gupta A., and Jain R. Content-based image retrieval at the end of the early years. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 22, 12* (Dec. 2000), 1349–1380.

[21] Tao Y. and Grosky W. Object-Based image retrieval using point feature maps. *Proc. of the Intl. Conf. on Database Semantics (DS-8)*, 1999, 59-73.

[22] Tuytelaars T., Van Gool L., Content-Based Image Retrieval Based on Local Affinely Invariant Regions. *Proc. of the 3rd Intl. Conf. on Visual Inf. Syst. (Visual'99)*, 1999, 493-500.

[23] Weber R., Böhm K. Trading Quality for Time with Nearest Neighbor Search. *Proc. of the 7th Intl. Conf. on EDBT*, Konstanz, Germany, 2000.

[24] Weber R., Schek H., Blott S., A Quantitative Analysis of Performance Study for Similarity-Search Methods in High-Dimensional Spaces. *Proc. of the 24th Intl. Conf. on VLDB*, New-York, US, 1998, 194-205.

[25] Yamane Y., Hoshiai T., Tsuda H., Katayama K., Ohta M., Ishikawa H., Multi-Vector Feature Space Based on Pseudo-Euclidean Space and Oblique Basis for Similarity Searches of Images. *Proc. of the 1st ACM Workshop Computer Vision meets Databases (CVDB)*, 2004.