Visual Feature Mining for Adapting Query-by-Example over Large Image Databases

Anicet Kouomou Choupo, Laure Berti-Équille

Visual Feature Mining for Adapting Query-by-Example over Large Image Databases

Anicet Kouomou Choupo, Laure Berti-Équille
IRISA, Campus universitaire de Beaulieu
F-35042 Rennes Cedex, France
Phone: +0033-299847100 Fax: +0033-299847171
{akouomou,berti}@irisa.fr

Abstract

This paper proposes a strategy for automatic selection and scheduling of visual features as search criteria that are relevant to the query-by-example on a large still-image database. This method is composed of the two main steps: the indexing and the retrieval processes. The indexing step uses K-mean clustering and association rule discovery as dimensionality reduction techniques on the descriptor values. The retrieval step is designed to order the query execution plans for speeding up the content-based retrieval over the image database. At this step, query-by-example processing is adapted in order to propose instantaneous and intermediate results that are progressively merged together with the advantage, for the users, on one hand, not to wait until the whole database has been processed by similarity search and, on the other hand, to allow them to stop the current execution of the query without losing the first partial results. We evaluate our method by comparing query execution time and result quality with the sequential search on all the low-level descriptors available in the image database. Our experiments show that we can get a result similar to the final one in less than half the time of the sequential search, which is a promising result for optimizing content-based retrieval.

1. Introduction

Content-based retrieval in very large multimedia databases exploits the indexation of various intermediate descriptions of the contents whose computing costs can be considerable. For each media type (video, text, audio, image), the set of potentially relevant descriptors is often very large. Many descriptors are proposed in the literature and standardized (e.g., MPEG-7, MPEG-21 [9], TV AnyTime). A still-image database can be represented at the pixel level in various ways [16]: by local visual features or global descriptors of color, texture, layout or shape. However, for answering a query-by-example, it is sometimes useless to search the entire database by computing similarity distances over every available low-level feature, because the discrimination power of the features (and their filtering capacities) often depends on the size of the database (i.e., some features may be very discriminative for small databases and they may be advantageously used for filtering the results, but they lose their property when they are flushed in millions of images).

Another major drawbacks of content-based retrieval based on sequential similarity search is that the query must be carried out in an exhaustive way over the whole database, because of the poor efficiency of indexation scheme of very large databases of images described by high-dimensional features. The consequence is an unacceptable waiting time for the user, until all the images of the database are compared to the image-query with a similarity distance. Moreover, any stop during the query execution has, as a consequence, the loss of all the retrieval information due to the lack of recovery strategies, forcing the user to restart the query again.

Contrary to traditional databases where many works have been carried out on query optimization and more recently on adaptive query processing [7,12], very few works have been proposed on multimedia content-based retrieval optimization. In our paper, we propose a strategy for automatic selection and scheduling of the search criteria (i.e., the visual features) that are relevant to a given query-by-example on a large still-image database. We use techniques of clustering and discovery of association rules on low-level descriptor values as dimensionality reduction techniques specifically applied to query-by-example optimization. We also present how the query-by-example processing can be adapted to propose instantaneous and intermediate results which are

progressively merged, with the advantage, for users, on one hand, not to wait until the whole database has been processed for finding the most similar images to the submitted image-query and, on the other hand, to allow them to stop the current execution of the query. We evaluate our method on a still-image database by comparison with the query performance and result quality obtained by the sequential search over the whole set of available features.

The rest of the paper is organized as follows: in section 2, we present the related works using discovery of association rules for indexing and searching images. We present then, in section 3, our method of automatic selection of the search criteria, and then we formalize, in section 4, the progressive adaptation of query-by-example execution plans. In section 5, we describe the experiments that validate our approach. Section 6 offers concluding remarks.

2. Related works

Content-based retrieval of images consists in searching the nearest neighbors of a query-by-example among all the available images indexed by multidimensional descriptors in the database (e.g., by MPEG-7 image descriptors that are vectors with 7 to 84 dimensions). Indexation techniques that are either based on data partitioning (such as R-Tree [8]) or on space partitioning (such as k-d-Tree [2] or GridFile [14]) remain adapted to low-dimensional vectors. The performances are very quickly deteriorating when the dimension of features increases. New indexation scheme have to be defined and adapted to the vectors of high dimension as well as approximate searching methods to improve the query performance.

In this context, several methods exploit, among the data mining techniques [18], the discovery of association rules in the context of multimedia content-based retrieval [4,15,17,10]. Djeraba [4] uses an indexing technique that combines the construction of clusters and the determination of association rules. Several subgroups of images (clusters) are thematically classified with labels (such as "animal", "plant"...). The aim is to characterize and labellize these image groups by association rules. At the query time, an analysis of the query starting from the association rules drives the choice of the images subgroups for the sequential search. The work of Djeraba [4] aims at the improvement of the quality of results from a semantic perspective without merging the results from several visual descriptors and without considering the query performance. Moreover, the approach imposes the creation of a thesaurus for the semantic labellisation of the clusters.

In our approach, the number and the type of descriptors are not limited and no labelling is required, because our objective is clearly to technically improve the query performance at the pixel level with no ambition on the semantic level. At the indexing step, our strategy combines cluster computation and extraction of association rules. Thus, it is entirely non-supervised and is appropriate for very large image databases.

Since *Apriori* [1], many algorithms for discovery of association rules have been proposed in the literature, they are very effective for classical transactional and relational databases whose structure is well defined. But their application to the images requires some adaptations. An idea consists in transforming the image descriptions and in applying the traditional techniques of association rule discovery. An image can be identified as a transaction and the various regions that compose it as objects [15]. Under certain conditions, the repetition of objects may carry information. Zaïane et al. [17] proposed a definition of association rules with recurrent objects as the following implication:

$a_1P_1 \wedge a_2P_2 \wedge ... \wedge a_nP_n \Rightarrow b_1Q_1 \wedge b_2Q_2 \wedge ... \wedge b_mQ_m$

where P_i and Q_j (with $i \in [1..n]$ and $j \in [1..m]$) are image descriptors and a_i, b_j are integers indicating the number of occurrences of the descriptors. The *MaxOccur* algorithm described in [17] is an adaptation of *Apriori* for the extraction of association rules with recurrent objects.

In our approach, we propose an application of the association rules for indexing and querying by example still-image databases. Our approach lies within the scope of the transformation of multimedia data and the application of the traditional techniques of discovery of association rules. We use several MPEG-7 image descriptors and the images are gathered into clusters for each descriptor. We consider an image as a transaction and a cluster identifier of each descriptor as an attribute. The problem of recurrent objects does not arise since the clusters are identified in a single way for each descriptor. The main contribution of this paper compared to our previous work [10] is the introduction of progressive query-by-example in order to propose intermediate results that are progressively merged together until the end of the query execution over the whole database. This extends the recent work of Kiranyaz and Gabbouj [11] on progressive query which was limited to the periodic execution of the same sub-query on portions of multimedia databases (images or videos) with no emphasis neither on scheduling the data subsets to process, nor on merging results from heterogeneous descriptors. The scalability and the computational

complexity of the approach were not studied in [11] and the size of the databases or of the database portions targeted by the sub-queries were not mentioned. Our study focuses on theses aspects for improving the query-by-example performance without deteriorating the quality of results compared to the sequential search.

3. Features mining and scheduling

Let us now present the principle of our method of automatic selection of the search criteria for the query-by-example on large still image databases. It includes: 1) the indexing step, based on clustering and discovery of association rules between the visual descriptors, 2) the retrieval step, based on selection and use of rules in order to choose the first-rank descriptors as search criteria (i.e., the most relevant for the query in terms of result quality and the most efficient for the query performance).

3.1. Indexing process

Off-line indexing process is based on the organization of the still-image database by computing the set of descriptor values for the characterization of the images. The image database is then gathered into several clusters using a K-means-like algorithm for each type of computed descriptors. The principle of the algorithm is as follows: data are partitionned into *k* clusters. We determine the number of clusters (*k*) empirically (see section 5.1). Each cluster centroid is firstly initialized by one image chosen randomly. The algorithm assigns each image to the cluster which the centroid is the nearest. It calculates new centroids and repeats the operations until no image are re-assigned (convergence).

Consider the image database *B* with *N* images, noted $I_1, I_2, ..., I_N$. Let $D = \{d_1, d_2, ..., d_m\}$ be the whole set of *m* available descriptors. We note $(n_j(I_i), d_j)$ the cluster of the descriptor d_j which the image I_i belongs to and we note *B*, the image database such as:

$$B = \{I | I = \{(n_1(I),d_1),(n_2(I),d_2),...,(n_m(I),d_m)\}\}$$

We apply the *Apriori* algorithm [1] for discovering association rules on the image database *B* and for obtaining the implications between the clusters of descriptors. We worked with a database of 30411 still images coming from a photo agency and described by the following MPEG-7 descriptors: *ColorLayout (CLD), ScalableColor (SCD), HomogeneousTexture (HTD), EdgeHistogram (EHD)* and *RegionShape (RSD).*

At the end of the indexing process, we obtain a set of clusters for each descriptor d_i and a set of association rules computed by the *Apriori* algorithm which identifies the relations between different clusters of the database *B* such as the rules `r1` and `r2` as examples:

```
r1:(19,HTD)∧(0,SCD)->(29,CLD)<0.1,66.1>
r2:(23,RSD)∧(1,HTD)∧(23,EHD)
                    ->(8,CLD)<0.1,55.2>
```

where `<0.1,66.1>` are respectively the support and the confidence values of the rule `r1`, expressed as percentages (idem with the support and confidence `<0.1,55.2>` of `r2`). Intuitively, semantics of a rule *r* selected for the query-by-example is: the nearest neighbors of the image-query that belong to all the clusters that are on the left-hand side of *r*, probably belong to the clusters that are on the right-hand side of *r* too.

We then use the most relevant association rules to schedule the sequential search on the ordered list of clusters, starting first with clusters that have the best selectivity (defined in section 4.1.) and that are on the left-hand side of selected association rules.

3.2. Content-based retrieval process

When the user submits a query-by-example to the system, his goal is to find all the images similar to the image-query according to several visual criteria. The user may know how to choose the visual descriptors as search criteria. But generally, he does not have any idea of what are the most discriminative descriptors for his query. In many cases, it is useless to search images on all the features already computed in the database. Our goal is to determine the priority order on the descriptors (and on their clusters) that are the most relevant, especially in the context of progressive queries that can provide intermediate and instantaneous results during the query execution and before the end of the exhaustive processing of the database by similarity search.

Given an image-query *Iq*, the system computes, for each descriptor d_j, a measure of selectivity of each cluster. The selectivity measure of a cluster depends on the size and the density of the cluster, and on its closeness to the description of the image-query relative to a given descriptor. The end of what we call a searching phase corresponds to the presentation of an intermediate result. The system determines, for the given searching phase, the *Top M* clusters to start with the similarity search and retrieves the nearest neighbors of I_q.

For the query processing using the association rules, the strategy is to first ignore the clusters present on the right-hand side of the selected rules and their corresponding descriptors. In the previous example, consider that the rules `r1` and `r2` are selected, then the query execution will first ignore clusters #29 and #8 of the descriptor *CLD* because they are in the right-hand side of the rules.

4. Adaptating query-by-example processing

Content-based retrieval generally requires the exhaustive processing of the whole image database before providing a result to users. Our goal is to improve the query processing while proposing: 1) a strategy of query-by-example execution plans which defines an optimal scheduling of the clusters to process with the similarity search, 2) the possibility to provide to users instantaneous results which are progressively refreshed and merged during the query execution, 3) the possibility for users to stop the current query execution if they are satisfied with the last intermediate result retrieved by the system.

The general principle is presented in the Figure 1. The indexing process is represented by the two first steps consisting in (1) the computation of the descriptor values and the clustering and (2) the generation of association rules (such as the rule `R1: C33` $\wedge$ `C23 ->C12`). The retrieval process starts at the step (3), a query-by-example I_q is submitted at the initial time T_o, and it is decomposed into several sub-queries targetting specific portions of the database (i.e., the *Top M* clusters) selected by the selectivity measure, and the priorities given by the association rules extracted beforehand at the indexing step (4). In Figure 1, the cluster C_{33} is the nearest to $D_3(I_q)$, the description of the image-query relative to the D_3 descriptor and it belongs to the first association rule `R1` previously generated. In this example, considering that the clusters `C33` and `C23` have a good selectivity, they will be selected as the first-rank clusters to process by the first subquery. At step (5), the intermediate results are progressively merged, updated and refined during the total query execution time over the whole image database and sent to the user (at T_1, T_2, T_3). The advantage is to provide instantaneous results to the user during the query execution. Moreover, the user will be able to stop the current execution (at T_{stop}) keeping the previous intermediate results. When the whole database has been processed, the last instantaneous result is merged with previous instantaneous results and is sent as the final result to the user at T_f.

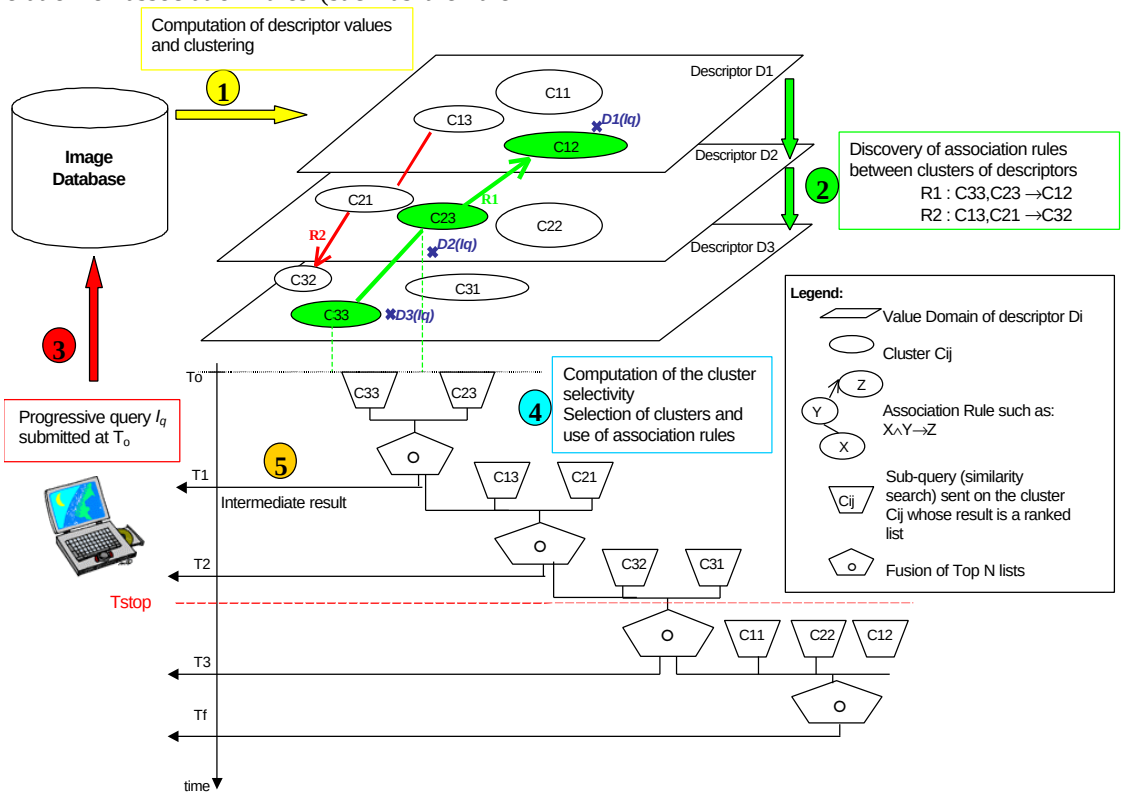

Figure 1. *Example of a progressive query execution plan*

4.1. Formal definitions

Image database. Let k be the number of clusters per descriptor and D the set of descriptors. The image database B is defined as the union of the k clusters C_{ij} of each descriptor d_i available in the database such as:

$$B=\bigcup_j C_{ij} \quad \forall d_i \in D, i=1..card(D), j=1..k \qquad (1)$$

An image belongs to the database if its description for each descriptor present in the database belongs to a cluster.

$$I \in B \ \ if \ \ \forall d_i \in D, \exists C_{ij}, d_i(I) \in C_{ij}$$

Query-by-example. A query-by-example $I_q = (n_q, O_q, W_q, D_q)$ for the multidimensional space S consists of the following information:

- The number n_q of points in I_q
- A set of n_q points $O_q=\{O_q^{(1)},\ldots,O_q^{(n_q)}\}$ in the d_S-dimensional feature space S
- A set of n_q weights $W_q=\{w_q^{(1)},\ldots,w_q^{(n_q)}\}$, the i[th] weight $w_q^{(i)}$ being associated with the i[th] point $O_q^{(i)}$ $(1 \geq w_q{}^{(j)} \geq 0, \sum_{i=1}^{n_q} w_q{}^{(i)} = 1)$
- A distance function D_q, which, given a point O in the space S, returns the distance between the query and the point. We assume D_q to be a weighted L_p metric, i.e., for a given value of p, the distance between two points T_1 and T_2 in S is given by:[1]

$$D_q(T_1,T_2) = \left[\sum_{j=1}^{d_s} m_q^{(j)} \left(|T_1[j] - T_2[j]| \right)^p \right]^{1/p} \qquad (2)$$

where $\mu_q{}^{(j)}$ denotes the weight associated with the i[th] dimension of S $(1 \geq \mu_q{}^{(j)} \geq 0, \sum_{j=1}^{ds} \mu_q{}^{(j)} = 1)$.

D_q specifies which L_p metric to use (i.e., the value of p) and the values of the dimension weights. We use the point distance function D_q to construct the aggregate distance function $D_q(I_q,O)$ between the multiple query points O_q and the point O (in S). $D_q(I_q,O)$ is the aggregate of the distances between O and the individual points $O_q^{(i)} \in O_q$:

$$D_q(I_q,O) = \sum_{i=1}^{n_q} w_q{}^{(i)} D_q\left(O_q^{(i)},O\right) \qquad (3)$$

[1] Note that this assumption is general since most commonly used distance functions (e.g., Manhattan distance, Euclidean distance, Bounding Box distance) are special cases of the L_p metric.

We use an equally weighted sum as the aggregation function but any other function can be used as long as it is weighted and monotonic.

Searching phase. The i^{th} searching phase corresponds to the similarity search processing over the set of i[th] rank clusters for each descriptor.

Selectivity of clusters. The selectivity of the cluster C for a given query-by-example I_q is a function combining the closeness of the cluster to the image-query (previously defined as the distance function D_q), the size and the local density of the cluster. Selectivity measure of a cluster C depending on the query-by-example I_q is noted *Selectivity(C, I_q)* and defined as follows:

$$Selectivity(C, I_q) = a\left(1 - \frac{dist(I_q, C)}{\max_j (dist(I_q, C_j))}\right) + b\left(1 - \frac{size(C)}{\max_j (size(C_j))}\right) + g \,.\, density(C) \qquad (4)$$

with $\alpha \geq 0, \beta \geq 0, \gamma \geq 0$ *and* $\alpha+\beta+\gamma=1$.

size(C) is the size of cluster C, *dist(I_q,C)* is the distance of I_q to the centroid of cluster C and *density(C)* is the density of cluster C at its centroid.
We evaluate the density using an index defined such as:

$$density(C) = \frac{\sum_{i=1}^{size(C)} \sum_{j=1}^{size(C)} |dist(I_i, C) - dist(I_j, C)|}{2 \times (size(C) - 1) \times \sum_{i=1}^{size(C)} dist(I_i, C)} \qquad (5)$$

with $I_i, I_j \in C$

dist(I_i,C) is the distance of I_i to the centroid of cluster C for $I_i \hat{I} C$. Parameters α, β, γ are positive and chosen such as *0 £ Selectivity(C, I_q) £ 1*. The priority for processing a cluster C is high for a given query-by-example I_q if *Selectivity(C, I_q)* is close to 1.

Association rule between clusters. An association rule r between several clusters of descriptors, noted C_{ij}, $C_{i'j'}$, $C_{i''j''}$, is defined as follows:

$$r: C_{ij} \wedge \ldots \wedge C_{i'j'} \rightarrow C_{i''j''} <supp, conf> \qquad (6)$$

with $i,i',i''=1..card(D)$, $i \neq i' \neq i''$ and $j,j',j''=1..k$

Semantics of the rule r is: an image whose description relative to the descriptors d_i and $d_{i'}$ belongs respectively to the clusters C_{ij} and $C_{i'j'}$ on the left-hand side of the rule r, probably belongs to the cluster $C_{i''j''}$ (on the right-hand side of the rule r), with *supp* and *conf* as support and confidence values of the rule r.

Selection of association rules between clusters. For a given query-by-example I_q, the clusters are selected depending on their selectivity measure at the i^{th} searching phase. For the i^{th} searching phase of I_q, the set of the *Top M* clusters is noted: $TopC_M^i(I_q)$.

The corresponding set of selected descriptors is noted: $TopD_M^i(I_q)$ after the rule selection.

An association rule is selected for query-by-example I_q, at searching phase k if its support and confidence are higher than given thresholds and if the set of clusters composing the rule, noted Cr, is such as:

$$Cr \subseteq \bigcup_{i=1..k} TopC_M^i(I_q) \quad (7)$$

When a rule is selected, the query processing will first ignore the clusters that are on the right-hand side of the rule.

Query-by-example execution plan. A logical query execution plan P for the query-by-example I_q is composed of the list of clusters C_{ij} to process for retrieving the most similar image according to all the descriptors D of the database. A query execution plan P for I_q is composed of several subplans whose results are merged with the fusion function noted "$\circ$" as follows:

$$P(I_q,D) \leftarrow P_1(I_q,d_1) \circ P_2(I_q,d_2) \circ \ldots \circ P_i(I_q,d_i) \circ \ldots \circ P_m(I_q,d_m) \quad (8)$$

$P_i(I_q,d_i)$ corresponds to the processing of the clusters for the descriptor d_i. There are $\prod_i (n_c(d_i))!$ possible plans with $n_c(d_i)$, the number of clusters of the descriptor d_i.

$P(I_q, D)$ can be rewritten with the list of clusters to process such as:

$$P(I_q,D) \leftarrow P_1(I_q,\{C_{11},\ldots,C_{1k}\}) \circ P_2(I_q,\{C_{21},\ldots,C_{2k}\}) \circ \ldots \circ P_m(I_q,\{C_{m1},\ldots,C_{mk}\})$$

A subplan can be rewritten such as:

$$P(I_q,\{C_1,\ldots,C_k\}) \leftarrow P_1(I_q,C_1) \circ \ldots \circ P_k(I_q,C_k)$$

with k clusters per descriptor and m descriptors.

Progressive query-by-example. A progressive query, noted $\vec{Q}(T_o, freq)$, submitted at the initial time T_o, and finished at final time T_f, is a query-by-example whose intermediate results produced by each subquery are regularly merged and sent to the user at $T_i \in]T_0,T_f[$ after the subquery execution depending on a frequency, noted *freq*. We suppose a partial order on the initial, intermediate and final times during the execution time of the query such as:

$T_0 < T_1 < T_2 < \ldots < T_f$.

Frequency of a progressive query. The frequency of a progressive query is the ratio of the required number of intermediate results and the number of minimal subqueries (i.e., the number of clusters). A minimal subquery corresponds to the similarity search over one cluster of descriptor. Frequency is defined by:

$$freq = \frac{Ni+1}{Nc} \quad (9)$$

with Nc : the number of minimal subqueries (i.e., the total number of clusters) and Ni : the number of required intermediate results ($0 \le Ni \le Nc-1$).

For the default frequency $freq = 1$, the intermediate results are sent to the user after the merging of the results of each minimal subquery. The corresponding progressive query on n clusters is:

$$\vec{Q}(T_0,1) = Q_1(T_1) \circ Q_2(T_2) \circ \ldots \circ Q_n(T_f)$$

In Figure 1, the frequency is $freq = 4/9$, with 3 intermediate results and 9 minimal subqueries (i.e., 9 clusters).

Instantaneous result. An instantaneous result of a progressive query $\vec{Q}$, is the result of a subquery on a portion of the database (i.e., one or a list of clusters of descriptor). It is obtained by the fusion of the previous intermediate results.

Final Result. The final result of a query is obtained:

- when the whole image database has been processed to answer to the initial query-by-example (i.e., when all the clusters have been processed by similarity search),
- when the query execution has been volontarily stopped by the user.

4.2. Generation and selection of query plans

Among all the possible query execution plans, query processing selects an optimal query execution plan that is executed by the query engine, which implements an execution model and produce the query result. Bushy plans offer the best opportunities to minimize the size of intermediate results. Hence, we represent query-by-example execution plan as an operator bushy tree. Nodes of the tree represent atomic physical operators and edges represent dataflow. A blocking edge indicates that the data is entirely produced before it can be consumed and an operator with a blocking input must wait for the entire operand to be computed and materialized before it can start. A pipelinable edge indicates that data can be consumed "one vector of descriptor values at a time". Therefore,

a pipelinable input (represented as a dotted line) can be consumed as soon as it has been produced. Figures 2a and 2b show the two cases of query execution plan corresponding to the query-by-example over the nine clusters of three descriptors given as example in Figure 1: the case of a traditional sequential search (Figure 2a) and the case of progressive query including cluster selection and association rules (Figure 2b). Two main classes of operators are considered in these plans: 1) binary symmetric operators that have one blocking inputs and one pipelinable input (such as *rank* for the same descriptor) or two blocking inputs (such as *fusion* for heterogeneous descriptors), and produce a pipelinable output, 2) unary operators (such as *scan*, *mat*). We introduce the unary operator *mat* in the query plan before each blocking edge to indicate that the materialization of intermediate results can occur at that point (in memory or on disk).

The first plan (Figure 2a) takes sequentially all the clusters of each descriptor before returning the result. It's written such as:

P1(d1,d2,d3) <- (C11**o**C12**o**C13**o**C21**o**C22**o**C23**o**C31 **o**C32**o**C33)(T_t)

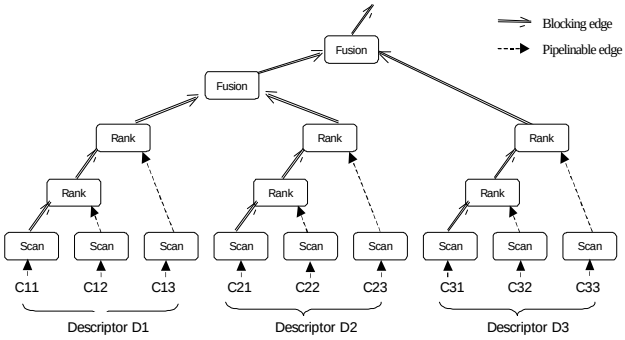

Figure 2a. Query plan tree for sequential search

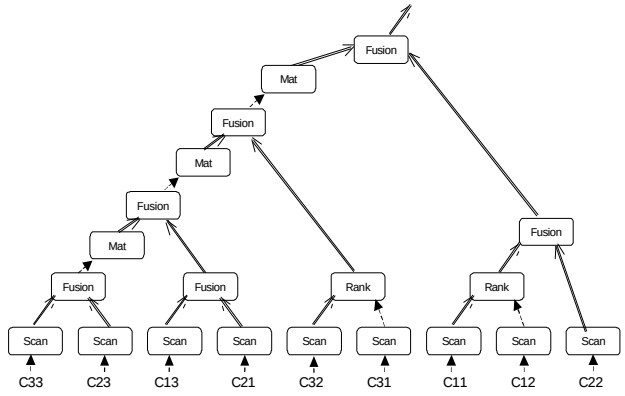

Figure 2b. Query plan tree for progressive query

The second plan (Figure 2b and also represented in Figure 1) is driven by cluster selectivity and discovered association rules and it returns three intermediate materialized results before the end of the query execution over the whole database.

It's written such as:

P2(d1,d2,d3) <- (C33**o**C23)(T_1) **o** (C13**o**C21)(T_2) **o**(C32**o**C31)(T_3)**o**(C11**o**C22**o**C12)(T_4=T_t)

Our goal concerning the execution strategy is to interleave the processing or the (partial) materialization of data from several clusters by selecting, scheduling and executing several query fragments concurrently in order to minimize the response time.

Consider the query-by-example I_q submitted to the image database and P_Q the set of all possible plans for the query-by-example, the search space for retrieving the N top plans is the set of size N of all the subsets of P_Q.

$$S(Q) = \{ P' \subseteq P_Q \mid |P'| = N \}$$

To retrieve the top N plans including the cluster selection step, we are currently adapting the Branch & Bound algorithm [13]. And we introduce the notion of upper bound limit for the plan P defined by the function *upperbound(P)* such as :

$$upperbound(P) \geq \max_{P' \supseteq P}[selectivity(P')] \qquad (9)$$

4.3. Fusion of intermediate results

For a given searching phase, the similarity search is first done over a selected subset of descriptors $\{d_1,d_2,\ldots,d_{m1}\} \subseteq D$ with $m_1 \leq card(D)$. Consider l_j, the list of query results for the descriptor d_j ranked in the decreasing order of similarity distance ($1 \leq j \leq m_1$). For computing the final result by merging the different ranked lists of results per descriptor, we define a merging score for the image I such as:

$$S_f(I) = \frac{1}{2m_1}\left(\sum_{j=1}^{m_1}\left(f_{d_j}(I) + S_{d_j}(I)\right)\right) \qquad (10)$$

with $S_{d_j}(I) \in [0,1]$ and $f_{d_j}(I) = \begin{cases} 1 & if \quad I \in l_j \\ 0 & \text{if} \quad I \notin l_j \end{cases}$

$S_{d_j}(I)$ is the score of I for the descriptor d_j. It is proportional to the similarity distance between the query-by-example I_q and I. Similarity between two images is the distance between the respective multidimensional vectors of the descriptors. We use the distances proposed by MPEG-7 in the definition of the descriptors. Merging *Top N* lists from heterogeneous descriptors (with different vector dimensions) is a difficult problem and we study other fusion functions [3,5] to implement in our experiments.

5. Experimental results

The method of automatic selection and ranking of the visual descriptors as search criteria for the query-by-example processing presented above is implemented in C++ on Linux. We're currently working with a database of 30411 heterogeneous still images coming from a photo agency. We will soon consider the processing of several hundreds of thousands of images.

At the indexation step, we exploit several MPEG-7 descriptors for color (*ColorLayout*, *ScalableColor*), for texture (*HomogeneousTexture*, *EdgeHistogram*) and for shape (*RegionShape*). For each MPEG-7 descriptor, we gather the images into clusters with a K-means-like algorithm. In this work, we observed that very large clusters (containing many images) mostly appear on the right-hand side of the association rules in order to keep the level of confidence high. The determination of the number *k* of clusters is difficult. Compared to other works [6], we adopt an experimental approach in which the homogeneity of the size of the clusters determines the choice of the number of clusters to consider. That is, we do not fix arbitrarily the number of clusters but we determine it depending on the image database population and on the number of robust association rules we want to consider (see Experiment 1).

At the retrieval step, we study the contribution of association rules (see Experiment 2) and the selection of clusters based on the selectivity measure (see Experiment 3). In these experiments, we use 100 images as queries-by-examples, each belonging to the database and we search for the *Top 15* nearest neighbors. We measure the average time and the average result quality. In our approach, the result quality at each searching phase for a given progressive query-by-example is defined as the number of common images, present both in the intermediate result and in the result obtained at the end of the sequential and exhaustive search over the whole set of descriptors of the image database. In other words, the quality of result is optimal when the quality is the same as obtained by sequential search (i.e., retrieving the same images). In this measurement of quality, we are not focused on the order of the retrieved images belonging to the result list but only on their presence/absence that could be satisfactory enough for the end-user at each searching phase.

5.1. Experiment 1: choosing indexing parameters

The goal is to study the parameters of indexing such as the number of clusters to fix, the thresholds of the support and of the confidence. First, we calculate, for a given value of the number of clusters, a quantity characterizing the distribution of the size of clusters around the average size. The higher this quantity is, the more the size of clusters is far from the average. Figure 3 shows the variations of the computed distribution against the number of clusters for each of the five descriptors: *ColorLayout*, *ScalableColor*, *HomogeneousTexture*, *EdgeHistogram* and *RegionShape* (respectively noted *CLD*, *SCD*, *HTD*, *EHD*, and *RSD).*

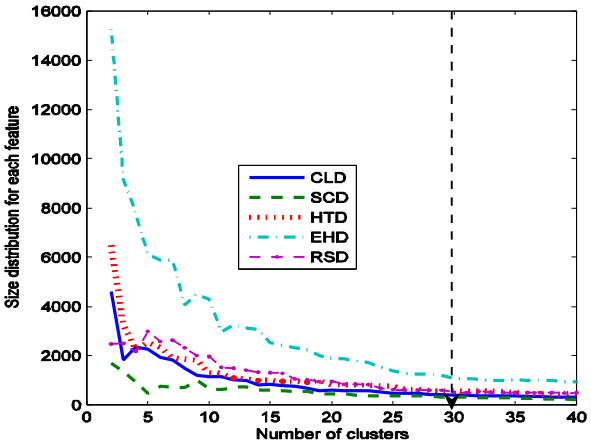

Figure 3. *Indexing parameters: the number of clusters*

We observe that, when the number of clusters is less than 10, the clusters are very disparate. For more than 20 clusters, the size of clusters significantly approaches the average. Hence, the number of cluster must be sufficiently large so that the size does not deviate significantly from the average. This is true for all the descriptors since they have the same distribution of the images in the clusters. For this reason, we can choose the same number of clusters for all the descriptors and, for this database, we fixed the number of clusters for each descriptor to 30 clusters per descriptor (Figure 3). The homogeneity of the size of the clusters is an important parameter that has an impact on query time over the whole set of clusters.

The association rules are computed over the clusters of descriptors with the algorithm *Apriori* [1]. The threshold of confidence is fixed at 50 % to guarantee the robustness of the rules. The threshold of the support is chosen experimentally. We obtain rules with very weak support in general. A high number of clusters involves the reduction of the support of the rules. Suppose that all the 30 clusters contain exactly

the same number of images (i.e. $\frac{30411}{30}$), then, under the assumption that the right and left-hand sides of the rules contain the same images, the support of a rule will be, in the best case, about $\frac{\left(\frac{30411}{30}\right)\times 100}{30411} \approx 3{,}33\%$.

The rules give nevertheless information about the number of images that are found in the same cluster for several descriptors. In the following, a rule will be selected if its confidence is at least 50 % and if the left and right hand sides of the rule have at least 10 common images. Our goal is then to manage a big number of rules with a fixed threshold of the support of 0,033 %. Under these conditions, 30 clusters numbered from 0 to 29 are computed for all the descriptors and 279 rules are produced at the end of the indexing process.

5.2. Experiment 2: using association rules for progressive query

The similarity search is first made on clusters of descriptors ranked according to the closeness of their centroid to the given image-query I_q. At the first searching phase, the query processing of I_q is done on the first-rank clusters (i.e., $TopC_M^1(I_q)$). At the next searching phase, the query processing is done on the second-rank clusters and the query results are merged with the ones of the previous searching phase and updated.

The database is organized into 30 clusters for each descriptor, and the query processing will thus have 30 searching phases. A progressive query means that intermediairy results can be sent at the end of each phase. We propose in this experiment to explore the strategy of the dynamic use of association rules: the rules are selected while the query is executed.

This approach requires the browsing of the whole set of association rules every time a query is submitted to the system. At the 12th searching phase, the quality of the intermediate result obtained by the dynamic use of association rules is almost the same as the final result obtained by sequential search. Figure 4 shows that, when we dynamically use association rules, the relative loss of quality is about $\frac{1\times 100}{15} \approx 6{,}67\%$. And the relative gain of performance (on query time) is about $\frac{(4{,}6-3{,}55)\times 100}{4{,}6} \approx 22{,}9\%$.

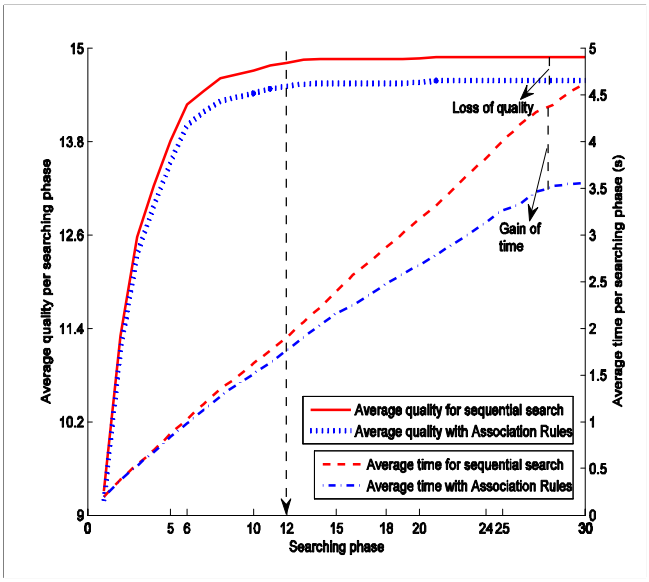

Figure 4. *Dynamical use of association rules*

The main advantage of progressive query execution planning is to retrieve the final result (with approximatively the same quality) after the half of the global query time necessary to a sequential search.

5.3. Experiment 3: selecting and ordering clusters

We study the measure of selectivity to show the importance of the order of clusters in the context of a progressive query. This measure combines the distance of the cluster centroid to the image-query, the size of the cluster and the density of a cluster at its centroid. It privileges the smallest, the nearest, and the most concentrated clusters at their centroid.

Based on the definition of the selectivity measure (previously given in section 4.1, formula (4)), for α = 1, β = 0 and γ=0, clusters are ranked according to the closeness of their centroid to the query image and we have the same results shown in Figure 4.

Curves of result quality and query time for different values of α, β, γ are respectively shown in Figure 5 and Figure 6.

The values α = 0, β = 1, γ=0 imply the clusters selection exclusively on the size criterion (Figure 5). In this case, the time grows slowly, but convergence of the result is slowed compared to the case (α = 1, β = 0, γ=0) of selection on closeness. We observe on Figures 5 and 6 that at equal time duration, partial result convergence speed is better for (α = 1/4, β = 3/4, γ=0).

We can then say that cluster selection on the size criterion is good for reducing time but the closeness criterion contributes to keep partial result convergence

high (i.e., with relatively good quality compared to sequential search). Partial result convergence is faster with (α = 1/3, β = 1/3, γ=1/3) than with (α = 1/4, β = 3/4, γ=0) (Figure 5), corresponding times varying similarly (Figure 6). This observation confirms that additional density-based criterion can speed up result convergence keeping retrieval time relatively low.

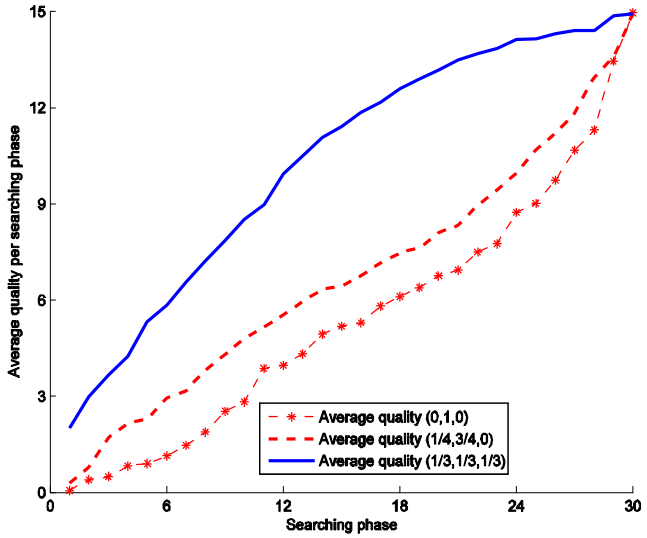

Figure 5. *Average result quality with differents parameters* α, β, γ *of selectivity*

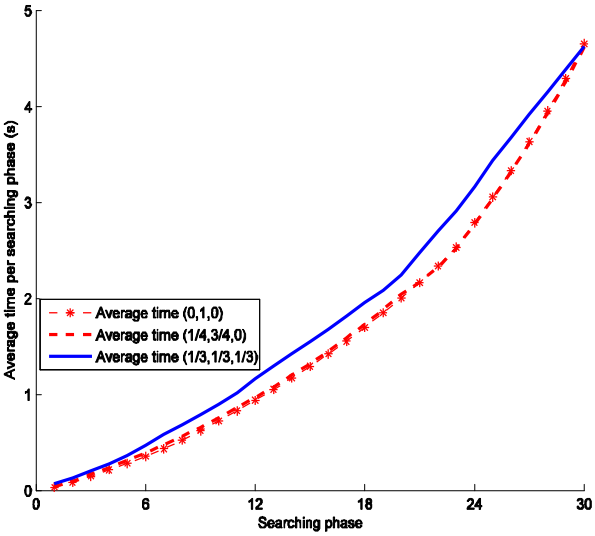

Figure 6. *Average query time with differents parameters* α, β, γ *of selectivity*

We conclude that using the selectivity measure for the cluster selection during the progressive query execution planning can improve the intermediate result quality.

5.4. Discussion

The previous experimental section presented the work done for the test and the validation of our method. The goal of the first experiment was to study the main indexation parameters that are: the number of clusters, the thresholds of the support and the confidence. The number of clusters was chosen to be the same for all features. The discovery of association rules can however be done, each feature having a different number of clusters, with no effect on the retrieval process. The method is then easily usable in the general case where the number of clusters may be different for each descriptor. We show in this experiment that thresholds of the support and the confidence essentially depend on the number of association rules to manage. Confidence threshold can be arbitrarily chosen but the more support threshold is reduced, the more we have rules and the less these rules are statistically significant.

The focus of the second experiment is set on using association rules in the scope of progressive queries. We show on one hand that, given a query-by-example, it is not necessary to process all the image database before retrieving results and, on the other hand, that the retrieval process is sped up using association rules. In this experiment, all queries-by-example are randomly taken in the image database and the results are compared to the results obtained by sequential search. The next step of validation of the result quality will involve users in the relevance feedback.

Efficient processing of progressive queries-by-example implies to know by which cluster to start. For this reason, it is then important to design several criteria leading to the definition of the priority order for processing the clusters for answering to a given query-by-example. The third experiment shows how the order of cluster processing plays an important role on the quality of intermediate results and on the response time. We work on a family of query plans in which we assume, at each searching phase, that one cluster is processed for each descriptor. We then consider implicitly that all features have the same number of clusters. It is completely possible to make the retrieval process more flexible by giving an order on clusters all together (whatever descriptor) with the advantage to start by clusters of the most interesting features for the query.

6. Conclusion

In this paper, we present an automatic strategy for selecting the search criteria for content-based retrieval over still-image databases. This strategy is based on the use of clustering and discovery of association rules techniques in order to set the priority on the relevant descriptors for a query-by-example and also to schedule the first-rank clusters to process. We introduce the measure of selectivity of clusters that we use to reduce the query time while keeping a suitable speed of convergence of the intermediate results close to the quality of results obtained by sequential search. All our experimental results are compared on the quality and the search time obtained with a sequential and exhaustive search on a database of 30411 still images described by five MPEG-7 descriptors. Our experimental results show that, on one hand, a strategy of progressive query execution plan makes it possible to have almost the totality of the final result (with the same quality) after half of the total query time of a sequential search and, on the other hand, clustering and discovery of association rules over clusters of images accelerates the retrieval process. The use of association rules is thus a promising technique for the cluster selection and the improvement of the query performance.

The perspectives of our work are now to prove the scalabitlity of our approach for indexing and querying very large images databases with several hundreds of thousands of images. In this context, we may have to refine our measurement of selectivity and add other measures for evaluating the retrieval cost. The size of clusters should be fixed to better manage I/O problems in the case of very large image databases (not only stored in main memory). The complexity of determination of clusters with K-means algorithm is quadratic with the size of the database, so we will have to look for more efficient methods of clustering.

7. References

[1] Agrawal R., Imielinski T., Swami A., Mining association rules between sets of items in large databases, *Intl. Conf. ACM SIGMOD*, 1993, p.207-216.

[2] Bentley J. L., Multidimensional binary search in database applications, *IEEE Transactions on Software Engineering, vol. 4, n° 5*, 1979, p. 333-340.

[3] Berretti S., Del Bimbo A., Pala P., Ischia, Merging results for distributed content-based image retrieval, *Intl. Workshop On Multimedia Information Systems*, 2003.

[4] Djeraba C., Association and content-based retrieval, *IEEE Transactions on Knowledge and Data Engineering, vol. 15, n° 1*, 2003, p.118-135.

[5] Fagin R., Kumar R., Sivakumar D., Efficient similarity search and classification via rank aggregation, *Intl. Conf. ACM SIGMOD*, 2003, p.301-312.

[6] Fernandez G., Meckaouche A., Peter P., Djeraba C., Intelligent Image Clustering, *Lecture Notes in Computer Science, vol. 2490*, 2002, p.406-419.

[7] Gounaris A., Paton N., Fernandes A., Sakellariou R., Adaptive query processing, *Nat. Conf. BNCOD, Lecture Notes in Computer Science*, vol. 2405, 2002, p.11-25.

[8] Guttman A., R-trees: A dynamic Index Structure for Spatial Searching, *Intl. Conf. ACM SIGMOD*, 1984, p.47-57.

[9] Kosch H., *Distributed Multimedia Database Technologies Supported by MPEG-7 and MPEG-21*, CRC Press, 2003.

[10] Kouomou-Choupo A., Berti-Équille L., Morin A., Multimedia Indexing and Retrieval with Features Association Rules Mining. *IEEE Intl. Conf. on Multimedia and Expo*, 2004.

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

[12] Manolescu I., Adaptive and self-tuning query processing, *EDBT Summer School*, 2002.

[13] Narendra P. M., Fukunaga K., A branch and bound algorithm for feature subset selection, *IEEE Transactions on Computers*, September 1977, p.917-922.

[14] Nievergelt J., Hinterberger H., Sevcik K. C., The GridFile: An adaptable, symmetric multikey file structure, *ACM Transactions on Database Systems, vol. 9 n° 1*, 1984, p.38-71.

[15] Ordonez C., Omiecinski E., Discovering association rules based on image content, *IEEE Advances in Digital Libraries Conf. (ADL'99)*, 1999, p.38-49.

[16] Smeulders A.W.M., Worring M., Santini S., Gupta A., Jain R., Content-based image retrieval at the end of the early years, *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, n° 12*, 2000, p.1349–1380.

[17] Zaïane O., Han J., Zhu H., Mining recurrent items in multimedia with progressive resolution re?nement, *IEEE Intl. Conf. on Data Engineering*, 2000, p.461-476.

[18] Zhang J., Hsu W., Lee M. L., Image mining: issues, frameworks and techniques, *Intl. Workshop on Multimedia Data Mining*, 2001, p.13-20.