Image Query Processing Based on Multi-level Signatures

F. Rabitti and P. Savino

IEI-CNR, Pisa - Via S. Maria 46, 56126 Pisa, Italy

ABSTRACT

This paper describes the processing of queries, expressing conditions on the content of images, in large image databases. The query language assumes that a semantic interpretation of the image content is available (i.e. an image symbolic interpretation), as result of an image analysis process. The image query language addresses important aspects of the image interpretations resulting from image analysis, by defining partial conditions on the composition of the complex objects, requirements on their degree of recognition, and requirements on their position in the image interpretation. Particular emphasis is given on the definition of suitable content-based access structures to make more efficient the query processing. An approach based on multi-level signatures is adopted. The query is pre-processed on the signatures to filter-out most of the images not satisfying the query. Finally, an evaluation of the efficiency and precision of the signature technique is given.

1. Introduction

The growing amount of electronically stored digital images has determining the need of systems able to efficiently manage large database of images. What is happening today is in some way similar to what happened in the sixties, when the growing amount of electronically stored data (mainly in the form of formatted records for business applications), to be managed efficiently in computer systems, led to the development of Data Base Management Systems (DBMS). A strong interest on image retrieval originated in the image processing community. Recent research activities concerning this topic (i.e. visual databases) are reported in [Kuni89]. A key aspect in image retrieval is to define suitable query languages and access structures [Chan81].

Different query approaches are described in [Choc89] for low-level image retrieval, in [Rose84] for image retrieval in CAD/CAM systems, in [Tang81] for alphanumeric and image data retrieval, in [Rabi89a] for graphical image retrieval. However, most of the image database systems proposed are special purpose systems. This means that the way images are stored, organized and retrieved depends on the application for which the system has been studied and cannot be generalized to different applications [Rabi89b].

In this paper we intend to present a general purpose image query language, and then describe the processing of queries in such a language. The starting point, i.e. the result of the image analysis process, can be found in [Rabi90] and [Rabi91]). In Section 2 a brief description of the Image Analysis is given, while the Query Language is introduced in Section 3. The complete query processing algorithm is given in Section 4. A multi-level signature technique, used to support an efficient image retrieval process, is then presented in Section 5. An evaluation of the efficiency and precision of this signature technique on image databases is also given. Final remarks are contained in Section 6.

2. Image Analysis

Image analysis may be accomplished only if the classes of images to be handled (i.e. the application domain) are determined and described in advance to the system. In this case, the images contain a finite set of objects with complex relationships among them. However, only the description of the application domain is specific of an application environment; other environments may require the definition of the domain(s) that are specific to them.

An image analysis process, based on a given application domain definition, tries to recognize the objects contained in the images, recording different interpretations, the associated degree of recognition and their position in the image space. Images may contain simple objects (hereafter called *basic objects*) and *complex objects*, which are composed of basic and complex objects. The analysis process tries to determine the composition of the

complex objects in terms of simpler objects. (Details on the automatic image analysis process can be found in [Rabi90] and [Rabi91].)

The result of the analysis process on an image is one ISR-DB (Image Symbolic Representation at Database Level, according to the format described in [Rabi90]) for each application domain selected. An ISR-DB can be empty if the image analysis process does not produce any acceptable image interpretation for the specific application domain.

In the following discussion we refer to a simple real-world example. The example belongs to an application domain composed of apartment's plants with their furnitures (*ApartmentDesign*). The image analysis allows the recognition of furnitures and types of rooms (e.g. dining room, bathroom, etc.) contained in the images. Furnitures are either basic objects (e.g. table, chair, etc.) or complex objects (e.g. double bed). Rooms are complex objects.

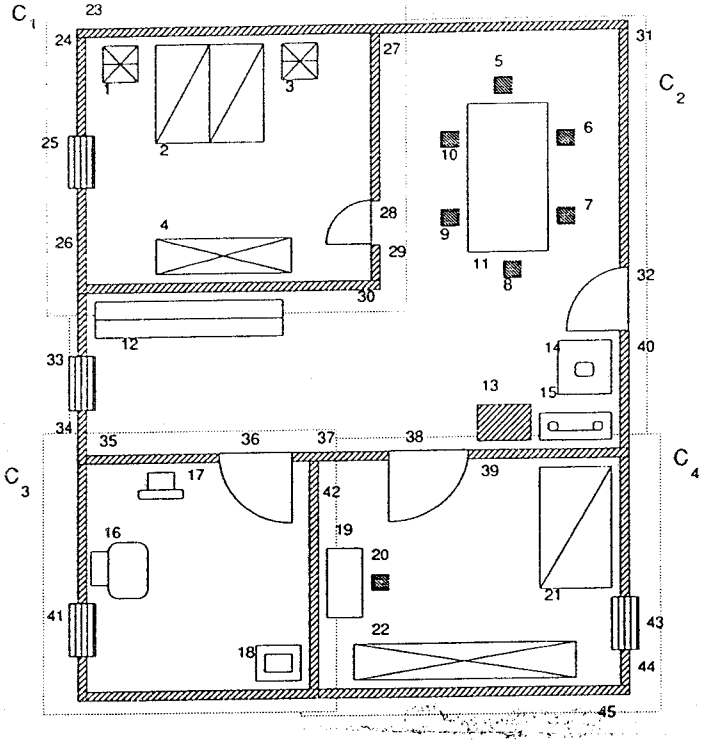

Figure 1 - Image example

3. Image Query Language

The image query language (described in detail in [Rabi91a]) allows to restrict the query to one or more application domains. Only the images, in the database, analyzed in the specified domain (or domains) will be considered. It also allows to express a Boolean combination (i.e. using AND, OR, NOT operators) of conditions (*object clauses*) on objects to be found in the various image interpretations. An object clause is expressed in *Conjunctive Normal Form*. Quantifiers (i.e. AT MOST, AT LEAST, EXACTLY) can be associated to an object specification. They serve to pose conditions on the number of object instances of the same object type to be found in the same image interpretation. Absolute positional constraints can be associated to an object specification and relative positional constraints can be specified between couples of object conjuncts.

Furthermore, an object specification can be nested, i.e. conditions can be given on the objects composing the required object (i.e. WITH followed by an object clause). This poses conditions on the rules exploited in the recognition of a specific complex object in the image. We must remember that in the application domain definition, several recognition rules can be associated to a specific complex object, leading to alternative object recognitions. The WITH clause allows the user to select one or more specific composition for a complex object in the image.

The main characteristics of this query language are illustrated in the following example:

Example 1:

```
FIND IMAGE IN DOMAIN ApartmentDesign
CONTAINING
  OBJECTS
  (DiningRoom RECOGN 0.7
      WITH (Kitchenette
            AND Table
            AND AT LEAST 4 Chair
            SUCH THAT
              ((OBJ(1), OBJ(2) ARE S),
               (OBJ(1), OBJ(3) ARE CLOSE)))
   AND EXACTLY 1 SingleBedroom
          RECOGN 0.7 POSITION (0.4, 0.7), (1, 1)
  );
```

The query clause is composed of only one object clause. The first object conjunct requires that in the image interpretation at least one complex object ("AT LEAST 1" is assumed if not otherwise specified) must be recognized as a *Dining Room* (O_{DIR}), with minimum recognition degree of 0.7.

The second conjunct requires that exactly one object must be recognized as a *Single Bedroom* (O_{SBR}) with minimum recognition degree of 0.7. This means that, while more objects O_{DIR} may be present in the image interpretation, only one object O_{SBR} must be present in the image interpretation to satisfy the object clause. Notice also the absolute positional constraint on the object O_{SBR}, requiring that the object be fully contained in the lower-right part of the image (i.e. within the rectangle determined by the points (0.4,0.7) and (1,1).

Example 1 also shows that the object clauses can be recursive. In general, a condition on the presence of an object (in an object conjunct or a disjunct) may contain another complete object clause (following the keyword WITH). This serves to pose further conditions on the

composition of an object, in terms of simpler objects (this also implies restrictions on the rule used to recognize an object). In Example 1, the first conjunct, requiring the presence of a *Dining Room* (O_{DIR}) has a further object clause associated. It means that, in order to satisfy this conjunct, it is not enough that any object O_{DIR} is recognized in the image interpretation, but it is necessary that O_{DIR} satisfies further conditions (i.e. the associated object clause). In particular, it requires that O_{DIR} must be composed, among the other objects, of (at least) one complex object *Kitchenette* (O_{KI}), (at least) one basic object *table* (O_t) and at least four basic objects *chair* (O_c). Moreover, two positional constraints are defined on the objects composing O_{DIR}: (1) OBJ(1), i.e. the object O_{KI}, must be positioned South (this is a way to specify directions in 2D images) with respect to OBJ(2), i.e. the object O_t; (2) OBJ(1), i.e. the object O_{KI}, must be *close* to OBJ(3), i.e. each of the objects O_c. Note that in our example application domain, we define two objects as *close* if the distance of their enclosing rectangles (on either X or Y axis) is lower than 1/4 of the image dimension (on the corresponding axis).

4. Image Query Processing

Here we describe a strategy for processing queries defined in the query language described in previous section. The symbols that will be used in the following discussion are summarized in Table 1.

Table 1. Summary of symbols used.

Symbol	Definition
I	Image
T	Image interpretation
C	Context
R	Context interpretation
Q	Query
QCNJ	Query conjunct
OC	object clause
OCNJ	Object conjunct
dom	Domain
SD(Q)	Set of application domains mentioned in Q
SO(Q)	Set of objects (basic and complex) mentioned in Q
L(dom)	Set of objects (basic and complex) of appl. domain
RDB	Set of all RSI-DB in the Image Database
RDB(dom)	Set of RSI-DB belonging to dom

The query is executed through the procedure *SearchImage*, which is composed of the following steps:

1) Parse query Q (a corresponding *parse tree* is generated).

2) The set of all domains specified in the query (D') is determined.
$D' = SD(Q)$

3) From the set D' all domains that are not compatible with the query, are removed. All domains that do not contain some of the objects mentioned in the query are eliminated. The resulting set D'' is as follows:

$$D'' = \left\{ dom \in D' \text{ such that } SO(Q) \subseteq L(dom) \right\}$$

4) Use the filtering technique, described in the following Section, in order to determine the query answer set

a) *filter–answer* $= \varnothing$
filter-answer is the query answer set determined on the base of the access methods adopted to speed-up the query processing. It contains a set of RSI-DB. It must be observed that each RSI-DB contained in *filter-answer* can be composed of a subpart of the corresponding RSI-DB in the image DB.

b) *for each* **dom** $\in D''$
filter-answer =
filter–answer $\cup$ *QueryFilter*(Q,dom)
where QueryFilter(Q,dom) is a procedure that returns the set of RSI-DB $\in$ RDB(dom) passing the filter corresponding to query Q, when executed on the image access structures adopted in this image retrieval approach. The procedure is described in the following Section.

5) Remove the false drops from the images contained in *filter-answer* and check for the positional constraints.

a) *answer* $= \varnothing$
answer is the query answer set

b) *for each* **RSI-DB** $\in$ *filter–answer*
if *QueryProc*$(Q,RSI\text{–}DB)$ *then*
RSI–DB $\rightarrow$ *answer*
where QueryProc(Q,RSI-DB) is a procedure that returns a boolean value. It returns **True** if Q is satisfied on RSI-DB, otherwise it returns **False**. The procedure QueryProc removes the false drops, verifies the positional and cardinal constraints and verifies again all query conjuncts. This procedure is described in [Rabi91a].

5. Access Methods

Here we describe the access structures that can support an efficient image retrieval process, based on the query language previously described. These access structures serve for fast access to the images in the database. Even if the access structures do not contain all the information in the image symbolic representations, it is sufficient that they restrict significantly the number of image headers to be accessed, where the complete query is performed to determine the final result of the query.

We propose here an access method based on a *signature technique*. The very idea of the *signature file access*

method is to extract and compress properties of data objects and store them in a separate file [Chri84]. The extracted pieces of data are called *signatures*. Also queries are supposed to be transformable to the signature form. A collection of the derived signatures is called signature file or *filter* because of its role during the query processing. The function of the filter is to determine all data objects which may qualify for a given query (i.e. to exclude data objects which do not satisfy the query). The signature file access method allows some *false hits* on the signature file level. (This is why a signature file is called "filter"). Therefore, a second step of query processing, called *false drop resolution*, is needed.

Table 2. Symbols used for signature structures.

Symbol	Definition
SIGN	Image signature
QSIGN	Query signature
f	Number of bits in the signature record
n	Number of bits set to "1" by a basic object
qw	Number of bits set to "1" in the signature for each object (query weight)

For each application domain, the signature of an object (simple or complex) is fixed as n specific bit positions in the complete signature block (f bits). The codes of all possible objects in the domain are specified in a look-up table, which may be updated to reflect the changes in the rules of image analysis process (e.g. rules for new objects or rules expressing more ways of recognizing the same object can be added). The value of n must be evaluated in relation to f, taking into account the average number of objects in each image, the number of different objects in the application domain, etc. to approach the target of 1/2 of ones, as average value, in the resulting signature block [Chri84]. A simple object (e.g. O_t in the example image) will set only n bits in the image signature, as specified in the application look-up table. A complex object, instead, will set its n bit position and the bit positions associated with all the simpler objects which compose it in the specific interpretation in the symbolic representation of the image. For example, the signature of O_{DIR} in the image example of Figure 1, is obtained superimposing the codes, obtained from the look-up table, of O_t, O_c, O_{KI}, O_{gs}, O_{kt}, O_{wb} and O_{sb}.

5.1. Access Structure Definition

We propose here a four-level signature (for each application domain in the system) for each image:

1) $SIGN(I)$ is the *image-level signature*. This signature is obtained by superimposing the codes of the image interpretations recognized in image I. More precisely, if the image is defined as

$$I = T_1, \ldots, T_n \quad (n \geq 1)$$

where T_i $(i=1,...,n)$ is a generic image interpretation, then

$$SIGN(I) = Sup_{i=1,...,n} \left[SIGN(I,T_i) \right]$$

so that SIGN(I) is defined in terms of SIGN(I,T), which is the *image-interpretation-level signature.*

Note that $Sup_{i=1,...,N}$ $(SIGN_i)$ is defined as the superimposition (i.e. bitwise OR) of the N signatures $SIGN_1, \ldots, SIGN_N$.

2) $SIGN(I,T)$ is the *image-interpretation-level signature.* This signature is obtained by superimposing the codes of the contexts recognized in interpretation T. More precisely, if the image interpretation T is defined as

$$T = C_1, \ldots, C_m \quad (m \geq 1)$$

where C_i $(i=1,...,m)$ is a generic context of the image interpretation T, then

$$SIGN(I,T) = Sup_{i=1,...,m} \left[SIGN(I,T,C_i) \right]$$

SIGN(I,T) is defined in terms of SIGN(I,T,C), which is the *context-level signature.*

3) $SIGN(I,T,C)$ is the *context-level signature.* This signature is obtained by superimposing the codes of the context interpretations recognized in the context C. More precisely, if the context C is defined as

$$C = R_1, \ldots, R_v \quad (v \geq 1)$$

where R_i $(i=1,...,v)$ is a generic context interpretation of the context C, then

$$SIGN(I,T,C) = Sup_{i=1,...,v} \left[SIGN(I,T,C,R_i) \right]$$

SIGN(I,T,C) is defined in terms of SIGN(I,T,C,R), which is the *context-interpretation-level signature.*

4) $SIGN(I,T,C,R)$ is the *context-interpretation-level signature.* The signature is obtained by superimposing the codes of the objects recognized in the context interpretation R of context C of image interpretation T of image I. More precisely, if the context interpretation R is defined as

$$R = O_1, \ldots, O_l \quad (l \geq 1)$$

where O_i $(i=1,...,l)$ is a generic object, either basic or complex, then

$$SIGN(I,T,C,R) = Sup_{i=1,...,r} \left[SIGN(O_i) \right]$$

The signature $(SIGN(O))$ of a generic object O is obtained through the following iterative procedure.
Let us distinguish two cases, one for the basic objects and one for the complex objects:

$$O = \begin{cases} case\ 1: & O_0 \ is\ a\ basic\ object \\ case\ 2: & O_0 \ is\ composed\ of\ O_1,...,O_r \ END \quad r \geq 1 \end{cases}$$

then

$$SIGN(O) = \begin{cases} case\ 1:\ code(O_0) \\ case\ 2:\ Sup\ \left[code(O_0), Sup_{i=1,\ldots,r}\ (SIGN(O_i)) \right] \end{cases}$$

where $code(O)$ is the code associated to the object O in the lookup table.

Therefore, the information contained in an ISR-DB is, for each image, as follows:

- one image-level signature;
- N image-interpretation-level signatures, where N is the number of different image interpretations in ISR-DB;
- M context-level signatures, with $M = \sum_{i=1}^{N} m_i$, where m_i is the number of different contexts in the *i-th* image interpretation;
- Z context-interpretation-level signatures, with $Z = \sum_{i=1}^{N} (\sum_{j=1}^{M} v_{i,j})$, where $v_{i,j}$ is the number of context interpretations in the *j-th* of the *i-th* image interpretation.

5.2. Filtering Process

The first step in the filtering process is to determine the signatures for the query. Two levels of signatures are used for the query. They are described in the following.

1) $QSIGN(Q)$ is the *query-level signature*.
A query Q is expressed as a conjunction of query conjuncts *QCNJ*:

$$Q = QCNJ_1\ AND\ \cdots\ AND\ QNCJ_M \quad M \geq 1$$

where four types of query conjuncts are allowed (we denote with OC an <object-clause>)

$$QCNJ = \begin{cases} type\ 1:\ OC \\ type\ 2:\ OC_1\ OR\ \cdots\ OR\ OC_p \\ type\ 3:\ NOT\ OC \\ type\ 4:\ OC_1\ OR\ \cdots\ OR\ OC_x\ OR \\ \qquad NOT\ OC_{x+1}\ OR\ \cdots\ OR\ NOT\ OC_{x+y} \end{cases}$$

$QSIGN(Q)$ is obtained by superimposing the signatures of the query conjuncts

$$QSIGN(Q) = Sup_{i=1,\ldots,M}\ \left[QSIGN(QCNJ_i) \right]$$

The signature of the query conjunct is expressed in terms of lower level signatures: the *object-clause level signature* $QSIGN(Q,OC)$.
Different signatures are generated, depending on the type of the query.

$$QSIGN(QCNJ) = \begin{cases} type\ 1: & QSIGN(Q,OC) \\ type\ 2: & Int_{i=1,\ldots,p}\ \left[QSIGN(Q,OC_i) \right] \\ type\ 3: & not\ defined \\ type\ 4: & not\ defined \end{cases}$$

Note that $Int_{i=1,\ldots,p}\ \left[QSIGN(Q,OC_i) \right]$ denotes the intersection (obtained by bitwise AND) of the p bit strings $QSIGN(Q,C_i)$, $i=1,\ldots,p$. Using the signature technique, the logical AND of search clauses is expressed by the *Sup* of the signature of the search clauses, while the logical OR of search clauses can be expressed by the *Int* of the signatures of the search clauses. A query signature resulting from a *Int* operation on several query signatures has a query weight (i.e. the number of bits set to "1") lower or equal to the query weight of the composing query signatures. The resulting query weight can even be zero (i.e. the query signature is made by all zeros).

It can be observed that the signature is not defined for queries of type 3 and type 4. Indeed, the signature method is not able to process queries containing the *NOT* operator. This is due to the fact that the signature method filters out records that are not relevant for the query, but not all of them. If the processing of a negative predicate (*NOT OC*) were done by searching for OC and selecting the items filtered out, some relevant items could be missed.

2) $QSIGN(Q,OC)$ is the *object-clause-level signature*.
A generic object clause OC has the following form

$$OC = OCNJ_1\ AND\ \cdots\ AND\ OCNJ_q \quad q \geq 1$$

where *OCNJ* is an object conjunct. Then

$$QSIGN(Q,OC) = Sup_{i=1,\ldots,q}\ \left[QSIGN(Q,OC,OCNJ_i) \right]$$

(For simplicity, we do not consider here object alternatives in the definitions of objects).

We will distinguish two forms of the object-conjunct (*OCNJ*): in one form the object conjunct has a WITH condition, while in another form it has no WITH condition.

$$OCNJ = \begin{cases} case\ 1: & O \\ case\ 2: & O\ WITH\ OC \end{cases}$$

where O is a generic object and OC is an object clause. Then

$$QSIGN(Q,OC,OCNJ) = \begin{cases} case\ 1:\ code(O) \\ case\ 2:\ Sup\ \left[code(O), recsign(OC) \right] \end{cases}$$

where $recsign(OC)$ is the signature record that corresponds to the object clause. $recsign(OC)$ is calculated using the following procedure:

$$recsign(OC) = Sup_{i=1,\ldots,q}\left[dsign(OCNJ)\right]$$

where

$$dsign(OCNJ) = \begin{cases} case\ 1:\ code(O) \\ case\ 2:\ Sup\ (code(O), recsign\ (OC)) \end{cases}$$

In Example 1, there is only one Type 1 conjunct in the query clause. Therefore, $QSIGN(Q)$ is equal to $QSIGN(Q,OC)$, i.e. the query-level signature is equal to the object-clause-level signature of the only object clause OC, contained in the Type 1 conjunct. There are two top-level objects in object clause OC, i.e. O_{DIR} and O_{SBR}. The signature of O_{DIR} is obtained by superimposing the codes of $O_{DIR}, O_{KI}, O_t, O_c$ (note that duplicates, such as O_c, are eliminated using the superimposed coding technique). The signature of O_{SBR} is equal to the code of O_{SBR}. Thus, $QSIGN(Q,OC)$ is obtained by superimposing the signatures of O_{DIR} and O_{SBR}.

The procedure *QueryFilter* is the central part of the query processing algorithm. It accesses the signature files of the domain *dom* and filters out all images whose signatures do not verify the query Q. The resulting set, $\overline{S}$ is returned. As already mentioned, the set $\overline{S}$ contains also some elements that do not match with the query Q (false drops). The algorithm returns, for each image in the result set $\overline{S}$ the corresponding RSI-DB. The RSI-DB of the images contained in $\overline{S}$ may correspond to a subpart of the RSI-DB of the same image in the DataBase. Indeed, it is possible that the query Q is verified only for some of the interpretations of the image, but not from all of them. The procedure *QueryFilter* is executed through the following steps:

Step 1: The image level signature file $SIGN(I)$ <*dom*> is scanned. $QSIGN(Q)$ is matched against the image-level signatures $SIGN(I)$ for all images I in the domain *dom*. The set S_1 is determined, defined as the set of all images whose $SIGN(I)$ matches $QSIGN(Q)$ (a query signature matches a data signature if all "one" positions in the query signature correspond to "one" positions in the data signature [Chri84]). S_1 is the first restriction of the image database. Obviously, if the query weight *qw* of $QSIGN(Q)$ is zero, this step is omitted and S_1 will contain all images in domain *dom*.

a) $S_1 = \emptyset$

b) *for each* $I \in RDB(dom)$
   if $(QSIGN(Q) \uparrow SIGN(I))$
   then $I \rightarrow S_1$
   else continue

Note: $(QSIGN(Q) \uparrow SIGN(I))$ means *match* the query signature $QSIGN(Q)$ with the image signature $SIGN(I)$. It returns *TRUE* if a match exists, *FALSE* otherwise.

Step 2: For each image I in S_1, determine the set of image-interpretation-level signatures $SIGN(I,T)$. If the query-level signature $QSIGN(Q)$ matches at least one image-interpretation-level signature $SIGN(I,T)$ and all query conjuncts of Type 1 and Type 2 are verified (according to rules (i) and (ii)), then I and the matching image interpretation T are inserted in the set S_2. S_2 is composed of couples (I,T). (i) A query conjunct of Type 1 is verified if the corresponding object clause has an object-clause-level signature that matches with the image-interpretation-level signature $QSIGN(Q) \uparrow SIGN(I,T)$. (ii) A query conjunct of Type 2 is verified if it contains at least one object clause with an object-clause-level signature that matches with the image-interpretation-level signature. S_2 is the second restriction of the image database.

a) $S_2 = \emptyset$

b) *for each* $I \in S_1$
   for *each* T_i *such that*
   $(QSIGN(Q) \uparrow SIGN(I,T_i)$ AND
   for each QCNJ (*query conjunct of type 1*)
   $(QSIGN(Q,OC) \uparrow SIGN(I,T_i))$ AND
   for each QCNJ (*query conjunct of type 2*)
   exists OC_j *such that*
   $QSIGN(Q,OC_j) \uparrow SIGN(I,T_i)$
   then $(I,T_i) \rightarrow S_2$
   else continue

Step 3: For each couple (I,T) in S_2, determine the set of context-level signatures $SIGN(I,T,C)$, corresponding to all contexts C of (I,T). If all query conjuncts of Type 1 and Type 2 are verified (according to rules (i) and (ii)), then (I,T,C) is inserted in S_3. S_3 is the third restriction of the image database. (I,T,C) is a triple that represents the context C of image interpretation T of image I. (i) At this level, a query conjunct of Type 1 is verified if the object clause that composes it is verified. An object clause is verified if its signature matches the context-level signature $(QSIGN(Q,OC) \uparrow SIGN(I,T,C))$. (ii) A query conjunct of Type 2 is verified if it exists at least one object clause that compose it that is verified.

a) $S_3 = \emptyset$

b) *for each* $(I,T) \in S_2$
   for each $C \in T$
   if *for each QCNJ* (*query conjunct of type 1*)
   $QSIGN(Q,OC) \uparrow SIGN(I,T,C)$ AND
   (*for each QCNJ* (*query conjunct of type 2*)
   exists $OC_i \in QCNJ$ *such that*
   $QSIGN(Q,OC_i) \uparrow SIGN(I,T,C))$
   then $(I,T,C) \rightarrow S_3$
   else continue

Step 4: For each context (I,T,C) in S_3, determine the set of context-interpretation-level signatures

$SIGN(I,T,C,R)$, corresponding to all context interpretations R of context C. If all query conjuncts of Type 1 and Type 2 are verified (according to rules (i) and (ii)) then (I,T,C,R) is inserted in $\bar{S}$. $\bar{S}$ is the result of the procedure *QueryFilter*. (I,T,C,R) is the context interpretation R of context C of image interpretation T of image I. (i) At this level, a query conjunct of Type 1 is verified if the object clause that composes it is verified. An object clause is verified if it has all object conjuncts with a signature that matches the context-interpretation-level signature $(QSIGN(Q,OC) \uparrow SIGN(I,T,C,R))$. (ii) A query conjunct of Type 2 is verified if it exists at least one object clause that composes it that is verified.

a) $\bar{S} = \varnothing$

b) *for each* $(I,T,C) \in S_3$
  for each $R \in C$
    if (*for each QCNJ* (*query conjunct of type 1*)
      $(QSIGN(Q,OC) \uparrow SIGN(I,T,C,R)$
) AND
    (*for each QCNJ* (*query conjunct of type 2*)
    exists $OC_i \in QCNJ$ *such that*
      $(QSIGN(Q,OC_i) \uparrow SIGN(I,T,C,R)$
)
    then $(I,T,C,R) \rightarrow \bar{S}$
    else continue

The result of the procedure *QueryFilter* is a set of RSI-DBs. They may not correspond exactly to the RSI-DBs of the images stored in the image database. Indeed, given a generic image I, having different image interpretations, each one composed of different contexts having different context interpretations, it is possible that only some of the context interpretations of I have a signature that verifies the query Q. The algorithm just described allows the selection of only these context interpretations. It must be observed that the RSI-DBs obtained as result of the procedure *QueryFilter* are directly compared with the query Q (in the step 5 of the procedure *SearchImage*). The possibility of reducing the complexity of the RSI-DBs used may reduce significantly the cost of this final step of the procedure *SearchImage*.

5.3. Evaluation

It is very difficult to perform an analytical evaluation of the signature-based access method to image databases proposed in this paper. The reason is that most of the simplifying assumptions used to analyze the signature techniques on text databases [Chri84] are not valid for image databases.

In particular, in text databases a very large number of distinct words (i.e. dictionary) is assumed (in real situations, it can be in the other of several hundred thousands [Rabi84]). Instead, in image databases, the distinct objects, basic and complex, are usually no more than few douzens. This is not a limitation on images in general but a limitation on the application domains on which the image analysis process can be applied.

Other assumptions on the frequency of words (i.e. geometric distribution of the frequency of word occurrency, low frequency of words on which to base the retrieval, etc.) cannot be simply extended to object images: no study exists on this topic but our experience (e.g. the *ApartmentDesign* application domain) is against these assumptions. Moreover, probabilistic independence of word occurrency, necessary to simplify the computations in case of text signatures, is unreasonable in case of image signatures, since semantic rules, used in the image analysis process, describe how object images can be combined to obtain more complex objects.

Therefore, most of the results obtained for signature techniques on text databases (e.g. the "optimal" ratio of "0" and "1" in signatures) cannot be extended to image databases. In order to get some insight in the performance of the proposed signature techniques on image databases, we decided to implement a prototype and measure the performance, in terms of space overhead and false drop ratio, on sample image databases. Obviously, we do not know how these results can be extended to other image databases (more experiments are necessary and a suitable analytical model, if possible, should be proposed).

The other main problem was that the number of images (about one hundred) that we had obtained from the image analysis process (in the *ApartmentDesign* application domain) was not significant to evaluate the performance of the retrieval technique. Adding new images to the database is very time consuming since it is necessary to draw (using a graphical editor) each new image before activating the image analysis process. Therefore, we decided to implement a *generator* of image symbolic representations (i.e. ISR-DB, as resulting from the image analysis), which should randomly generate representations of images "significant" in the *ApartmentDesign* application domain. This program has enbedded the knowledge of the rules for complex object recognition of our sample application domain, the knowledge of the context that may appear in these images, the combination of objects that are significant in the domain, etc.

This program has been parametrized, so to obtain (ISR-DB of) images with different degree of complexity (in terms of number of image interpretations, number of contexts in each image interpretation, number of context interpretations, number of complex objects and basic objects, not recognized as part of complex objects, in each context interpretation). With this *generator*, we have obtained three image databases: IDB_S (with simple images), IDB_M (with medium-complexity images) and IDB_C (with complex images) each one containing 10000 images. The characteristics of these image databases are

described in Table 3.

Table 3. Image database characteristics

	Im.Int.	Cont.	Cont.Int.	B.Obj.	C.Obj.
IDB_S	1-2	1-2	1-2	1-4	1-3
IDB_M	1-2	1-3	1-3	2-6	1-4
IDB_C	1-3	1-3	1-3	2-8	1-5

Then, we ran the prototype image retrieval system on these database to collect measurements on the 4-level signatures generated, and on the false drops resulting on a few sample queries. Table 4 describes the composition, in terms of "0" and "1" bits, of the signatures generated for each sample database. We tried four different values of f, i.e. the dimension of the signature block for each signature level. Note that the same block size is used at level 1, for the complete image, at level 2, for an image interpretation, at level 3, for a context, and at level 4, for a context interpretation. (In the following tables, SIGN_I means $SIGN(I)$, SIGN_T means $SIGN(I,T)$, SIGN_C means $SIGN(I,T,C)$, SIGN_R means $SIGN(I,T,C,R)$).

Table 4. Percentage of "1" in 4-level signatures

		IDB_S	IDB_M	IDB_C
f=64	SIGN_I	87.8	93.6	95.6
	SIGN_T	83.6	91.7	93.0
	SIGN_C	77.5	85.2	88.2
	SIGN_R	64.7	75.8	79.7
f=128	SIGN_I	70.5	79.4	82.6
	SIGN_T	65.9	76.1	77.9
	SIGN_C	55.6	66.0	70.0
	SIGN_R	46.4	53.0	57.7
f=192	SIGN_I	56.4	60.4	68.7
	SIGN_T	49.8	55.7	64.0
	SIGN_C	42.3	47.7	56.1
	SIGN_R	34.6	40.2	44.3
f=256	SIGN_I	47.3	56.7	59.5
	SIGN_T	41.0	53.3	55.1
	SIGN_C	34.2	43.6	47.1
	SIGN_R	27.5	32.2	36.0

We can note that, reasoning as in the case of text signatures, the percentage of "1" seems acceptable only for f=256, and for f=192 for the levels SIGN_C and SIGN_R.

Table 5 illustrates the memory overhead of the 4-level signatures, with respect to the different image databases. The overhead is computed as the ratio of the size of the signature blocks, at a specific level, referring to the same image, with the size of the ISR-DB of that image. The average size of an ISR-DB is 268 bytes, for IDB_S, 601 bytes, for IDB_M, 965 bytes, for IDB_C. The original image size has not been considered here since it depends greatly on the type of the original image. In our application domain, if the image is stored as a bit-map it would require about 500 KB (without compression), while if it is coded using graphical primitives (ZIP of Andrew, see the following section) it would require from 5 to 10 KB.

Table 5. Memory overhead of the 4-level signatures

		IDB_S	IDB_M	IDB_C
	ISR-DB size	2.68MB	6.01MB	9.65MB
f=64	SIGN_I	9.0	3.9	2.5
	SIGN_T	13.0	5.9	5.1
	SIGN_C	19.9	12.0	10.0
	SIGN_R	30.2	24.2	20.0
f=128	SIGN_I	11.9	5.2	3.3
	SIGN_T	17.3	7.9	6.8
	SIGN_C	26.6	16.0	13.4
	SIGN_R	40.2	32.3	26.6
f=192	SIGN_I	5.9		4.2
	SIGN_T	21.6	17.8	8.5
	SIGN_C	33.2	26.3	16.7
	SIGN_R	50.3	39.3	33.3
f=256	SIGN_I	17.9	7.8	5.0
	SIGN_T	26.0	11.9	10.2
	SIGN_C	39.9	26.7	20.0
	SIGN_R	60.3	48.4	40.0

We present now four sample queries, which have been used to evaluate the false drop ratio in our signature-based query processing techniques. Query 1 is the simplest: it requires images containing a complex object, DoubleBedroom, and specifies a particular interpretation of this complex object (i.e. a specific recognition rule) giving three objects that must be part of it. Queries 2 and 3 are both composed of the same two object clauses: in query 2 they are in AND, in query 3 they are in OR. Therefore, query 3 is the least selective query. Moreover, the weight (i.e. the number of "1") of its query-level signature is very low, since it is obtained intersecting its two object-clause level signatures. Query 4 is the most complex: it is composed of three query conjuncts, the last of them containing two object clauses in disjunction. We did not put quantifiers or cardinality constraints in the queries, since they cannot be verified using the signatures (we were interested to measure only false drops to the inherent signature unprecision).

Query 1:

```
FIND IMAGE IN DOMAIN ApartmentDesign CONTAINING
  OBJECTS (DoubleBedroom
       WITH (DoubleBed AND BedsideTable
             AND Wardrobe));
```

Query 2:

```
FIND IMAGE IN DOMAIN ApartmentDesign CONTAINING
  OBJECTS (DoubleBedroom
         WITH (DoubleBed AND BedsideTable
               AND Wardrobe)
       AND Kitchenette
         WITH (GasStove AND WashBasin));
```

Query 3:

```
FIND IMAGE IN DOMAIN ApartmentDesign CONTAINING
  OBJECTS (DoubleBedroom
         WITH (DoubleBed AND BedsideTable
               AND Wardrobe))
  OR OBJECTS (Kitchenette
         WITH (GasStove AND WashBasin));
```

Query 4:

```
FIND IMAGE IN DOMAIN ApartmentDesign CONTAINING
  OBJECTS (DoubleBedroom
        WITH (DoubleBed AND BedsideTable
              AND Wardrobe))
  AND OBJECTS (Kitchenette
        WITH (GasStove AND WashBasin))
  AND (OBJECTS (StudyRoom
        WITH (Table AND Chair))
     OR OBJECTS (DiningRoom
        WITH (Table AND Chair))
  );
```

We must observe that the query processing strategy that has been presented produces other errors, apart those deriving from the signature false drop. Indeed, it does not allow to distinguish between a query having a WITH condition and a query having the same objects as conjuncts in the object clause (O_1 *AND* O_2 *AND* O_3 and O_1 *WITH* (O_2 *AND* O_3) give the same result). We will not consider this error in the evaluation, since we are interested in measuring the false drop that derives from the use of the two level signatures. A different approach, based on more levels of query signatures [Rabi91a] removes this limitation.

Table 6 illustrates the false drop ratios, measured for the four queries, executed using the 4-level signatures on the three image databases, IDB_S, IDB_M and IDB_C. The signatures have been computed for four values of f: 64, 128, 192, 256.

These results show that the false drop ratio depends mainly on the type of image (i.e. on IDB_S, IDB_M, IDB_C) and on the type of query. Surprisingly, the factor f, which determine the memory overhead, does not have such a big impact on the false drop ratios. Some improvements can be noticed from f=64 to f=128, but very little, if any, from f=128 to f=192 and to f=256. Comparing Table 4 with Table 6, we can observe how the impact of signature weights (i.e. percentage of "1") on false drops, varying f, is not significant, and results from text databases cannot be generalized to our case.

More surprisingly, Table 6 shows that, even if false drop ratios are in general rather high (mainly for Query 3, the simple OR query, and for medium to complex images, IDB_M and IDB_C) when compared to theoretical values for text signatures (however, those values are computed for one word queries, with average word selectivity, and this selectivity cannot be compared with the selectivity of any image objects), the false drop ratio is zero (except in a few cases for f=64) at the last signature level, i.e. processing $QSIGN(Q,OC)$ on $SIGN(I,T,C,R)$. This means that the final false drop is lower than 10^{-4} (since the image database contains 10000 images). This result demonstrates the advantage of our multi-level signature query processing approach. In this approach, the resulting false drop ratio is the one obtained at the end of the forth step (and this seems to be surprisingly good). The previous steps are useful to limit the application of the final step on smaller portions of $SIGN(I,T,C,R)$, which, according to Table 5, is largest than the other three signatures combined. However, more complex query processing strategies could be studied: for example, applying selectively some of the first three query processing steps depending on the average complexity of images in the database, and the characteristics of the query to be processed (e.g. expected selectivity, etc.).

6. Final Remarks

In this paper, we have proposed an approach to the processing of content-based queries on image databases. The crucial point of this proposal is the strong integration of the image retrieval process with the result of the image analysis process.

The approach presented in this paper constitutes an improvement of the techniques used in the MULTOS prototype, with enhancements in the image analysis and retrieval process. (MULTOS is a prototype system for the filing and retrieval of multimedia documents [Cont90].) The image analysis process has been enhanced with the introduction of the concept of context and different interpretations of contexts and complex objects. The image retrieval process has been enhanced improving the query language and the query processing techniques.

For what concerns the retrieval process, we think that the ISR-DB seems an adequate starting point to study further the image retrieval process, even if substantial modifications or alternative approach in the image analysis process will be proposed in future. The signature-based query processing techniques needs further evaluation to understand better its behavior and directions for improvements. In particular, we are studying more flexible query processing algorithms, based on more levels of query signatures, and we are investigating how to combine spatial access methods (to process object positional requirements in the query) with these signature techniques.

REFERENCES

[Chan81] Chang N., Fu K., "Picture Query Languages for Pictorial Data-Base System", IEEE Computer Vol.14 (Nov. 1981)

[Chri84] Christodoulakis S. and Faloutsos C., "Signature Files: An Access Methods for Documents and its Analytical Performance Evaluation", ACM Transactions on Office Information Systems, Vol. 2, N. 4, pp. 267-288 (1984).

[Cont90] Conti P., Rabitti F., "Image Retrieval by Semantic Content", in "Multimedia Office Filing and Retrieval: The MULTOS Approach", edited by C. Thanos, North-Holland Series in Human Factors in Information Technology, North-Holland, 1990.

[Kuni89] "Visual Database Systems", edited by T. L. Kunii, North-Holland (1989).

[Rabi84] Rabitti F. and Zizka J., "Evaluation of Access Methods to Text Documents in Office Systems", Proceedings of 3rd Joint ACM-BCS Symposium on Research and Development in Information Retrieval, Cambridge, England, July 1984, pp. 21-40.

[Rabi89a] Rabitti F. and Stanchev P., "GRIM_DBMS: a GRaphical IMage Data Base Management System" Proc. IFIP TC-2 Working Conference on Visual Database Systems, Tokyo (April 1989), in "Visual Database Systems", edited by T. L. Kunii, North-Holland, pp. 415-430 (1989).

[Rabi89b] Rabitti F. and Stanchev P., "Image Database Management Systems: Applied Theories, Tools and Decisions" Proc. of the Third International Conference on Automatic Image Processing, CAIP-89, Leipzig, GDR (Sept. 1989), in "Computer Analysis of Images and Patterns", edited by K. Voss, D. Chetverikov, G. Sommer, Akademie-Verlag, Berlin, pp. 208-214 (1989).

[Rabi90] Rabitti F. and Savino P., "Image Analysis for Semantic Image Databases", IEI-CNR Tech. Rep. B8-34, Pisa, Sept. 1990.

[Rabi91] Rabitti F. and Savino P., "Automatic Image Indexation and Retrieval", Proc. RIAO'91, Barcelona, Spain, April 2-5, 1991.

[Rabi91a] Rabitti F. and Savino P., "Content Based Retrieval in Semantic Image Databases", IEI-CNR Tech. Rep. B9-12, Pisa, Febr. 1991.

[Rose84] Rosenthal A., Heiler S., Manola F., "An Example of Knowledge-Based Query Processing in a CAD/CAM DBMS", Proc. Tenth Int. Conf. on VLDB, Singapore, pp. 363-370 (1984).

[Tang81] Tang G., "A Management System for an Integrated Database of Pictures and Alphanumerical Data", Computer Graphics and Image Processing 16, pp. 270-286 (1981).

Table 6. False drops

f=64	IDB_S	I	T	C	R	IDB_M	I	T	C	R	IDB_C	I	T	C	R
	q1	0.28	0.23	0.18	0.088	q1	0.48	0.46	0.37	0.19	q1	0.55	0.52	0.45	0.29
	q2	0.25	0.17	0.08	0.014	q2	0.61	0.56	0.37	0.07	q2	0.74	0.69	0.54	0.16
	q3	0.76	0.15	0.07	0.011	q3	0.89	0.43	0.32	0.06	q3	0.91	0.5	0.4	0.12
	q4	0.27	0.07	0.02	0.0005	q4	0.64	0.46	0.22	0.01	q4	0.8	0.64	0.4	0.04
f=128	IDB_S	I	T	C	R	IDB_M	I	T	C	R	IDB_C	I	T	C	R
	q1	0.19	0.12	0.06	0	q1	0.40	0.35	0.22	0	q1	0.48	0.41	0.26	0
	q2	0.17	0.10	0.04	0	q2	0.50	0.43	0.22	0	q2	0.67	0.58	0.35	0
	q3	0.67	0.06	0.03	0	q3	0.84	0.19	0.11	0	q3	0.87	0.25	0.15	0
	q4	0.19	0.04	0.01	0	q4	0.55	0.35	0.12	0	q4	0.73	0.53	0.25	0
f=192	IDB_S	I	T	C	R	IDB_M	I	T	C	R	IDB_C	I	T	C	R
	q1	0.19	0.12	0.06	0	q1	0.40	0.35	0.22	0	q1	0.48	0.41	0.26	0
	q2	0.17	0.10	0.04	0	q2	0.50	0.43	0.22	0	q2	0.67	0.58	0.35	0
	q3	0.52	0.06	0.03	0	q3	0.74	0.19	0.11	0	q3	0.80	0.25	0.15	0
	q4	0.19	0.04	0.01	0	q4	0.55	0.35	0.12	0	q4	0.73	0.53	0.25	0
f=256	IDB_S	I	T	C	R	IDB_M	I	T	C	R	IDB_C	I	T	C	R
	q1	0.19	0.12	0.06	0	q1	0.40	0.35	0.22	0	q1	0.48	0.41	0.26	0
	q2	0.17	0.10	0.04	0	q2	0.50	0.43	0.22	0	q2	0.67	0.58	0.35	0
	q3	0.52	0.06	0.03	0	q3	0.74	0.19	0.11	0	q3	0.80	0.25	0.15	0
	q4	0.19	0.04	0.01	0	q4	0.55	0.35	0.12	0	q4	0.73	0.53	0.25	0