A Tool for Content-Based Image Retrieval in Object-oriented Databases

Caetano Traina Jr.[1]; Rildo R. dos Santos[2]; Agma J. M. Traina[1]; Edna Y. Senzako[1];

[1]Computer Science Department - Mathematics Institute at Sao Carlos of
Sao Paulo University - USP - Brazil
[2]Computer Science Dept - State of Sao Paulo University at S.J. Rio Preto - UNESP - Brazil
E-mail: [Agma|Caetano|Senzako]@icmsc.sc.usp.br , Rildo@dcce.ibilce.unesp.br

Abstract

Conventional image processing algorithms used to compare images are very time-consuming, making them inappropriate for use in searching for an image in a huge collection of images stored in a database. The usual approach uses icons and textual attributes stored with the images to specify the queries. However, this approach depends on human analysis of each image, which reduces the precision to a great extent. This work states that it is possible to quickly retrieve a set of characteristics from each image as it is being stored in the database, building an index structure based on each characteristic, so images can be retrieved through queries based on the graphic contents of the stored images. When a search for a specific image becomes necessary, these same characteristics can be extracted from that image, and the set of indexes can be used to narrow the search space of the target images progressively.

This work describes how these concepts were used to implement a tool that enables the user to issue commands to retrieve images stored in an Object Manager based on their contents. It is aimed at the storage of medical images as one kind of data associated to objects. The object-model underlying this tool defines the "image data type," a novel approach that permits a significant reduction of the set of images to be processed, based on their graphical contents, so the search can be accelerated when queries are issued.

Keywords: Image Database, Digital Library, Medical Databases, Image Information Systems, Data Modelling, Object-oriented Databases.

1. Introduction

Nowadays, computers are fundamental resources in all areas of human knowledge, providing information and giving tools to aid in its retrieval and analysis and supporting decision-making. Medicine is an area that uses increasingly more equipment to generate large amounts of data in digital format, much of which must be stored for long periods [Dutton-90]. A kind of equipment that has aroused much interest is the Magnetic Resonance Tomography (MRT) device [Vidoto-86], which has been used to obtain images from internal structures of the human body. The high quality images obtained can be cross-compared with other images, looking for predefined biological structures, in the search for the appropriate diagnoses. This and other modern medical equipment can generate large amounts of data that can be stored and indexed to be retrieved quickly later.

This paper assumes that these images are being stored into databases. However, the highly variable nature of this kind of data is usually inadequate for traditional Database Management Systems (DBMS) [Chang-92], such as those based in the Relational Model. A Database System suitable to support applications that must efficiently handle images must be able to effectively store, manipulate, retrieve and distribute images, as well as non-image data. Such systems are known as Picture Archiving and Communication Systems (PACS). When applied to store medical images, a PACS provides graphical information of the many aspects of each patient's health, diseases and treatment for the physician and technicians of a health care centre [Dutton-90]. Therefore, the database must store all the images in one place, including the parameters used to gather the image, the criteria and results of the diagnoses, etc.

Descriptive information is stored associated to images, such as comments and the diagnosis already made by a physician, or shortened icons of each image. Existing systems usually use this associated data to retrieve the images. The textual data is obtained with the help of a specialist, but the description depends either on his/her knowledge of each specific topic, or on his/her purpose at the time of analysis. Therefore, textual data cannot substitute the actual information contained in an image. An iconic image is also not appropriate for the domain of a medical image database because it is difficult to select an icon that could, in a significant way, represent the complex information presented in this image [Cardenas-93] [Gudivada-95]. Thus, meta-information cannot substitute the actual information contained in an image.

The search for stored images based on their full contents, although highly desirable, imposes a processing load on the search data engine that the current machines cannot work with within an acceptable period of time. This work describes an alternative way to deal with this excessive workload by pre-processing each image as it is being stored into an Object-oriented Database Manager (Object Manager) based on the SIRIUS Model [Biajiz-96], called SIRIUS/GO. The images are treated as a new data type, so index structures can be constructed over them. The index structures are built using a set of parameters of each image extracted from each image by an Image Matching System [Senzako-96]. Index structures can quickly reduce the set of target images to be retrieved when queries are issued.

This work is part of a joint project involving activities in the Physics and Computer ScienceDepartment, of the Physics Institute at Sao Carlos (IFSC), the Computer Science Department at the Sao Carlos Institute of Mathematics (ICMSC), both of the University of Sao Paulo (USP), and the Santa Casa Hospital in Sao Carlos [Traina-97]. This project consists of the development of a complete Magnetic Resonance (MR) Tomography System [Vidoto-86] [Chow-90]. This paper deals with the image storage and visualization system.

2. Image Constants, Image Summarizers, Image Parameters, and Correlated Concepts.

To be supported as native data types in an Object Manager, Images must be defined as part of its data meta-model. This work was developed using the SIRIUS data model as its underlying object model [Biajiz-96]. In SIRIUS, images are a specialization of the idea of Attributes being associated to Objects. SIRIUS assumes that every attribute has a **characteristic**, which defines the set of operations where it is involved. For example, a number is an attribute characteristic, so numbers are attributes that can be summed, multiplied, and so on; texts are attributes of other characteristics since texts can be searched for substring matching, can be concatenated, and so on. It must be noted that the set of operations allowable over an attribute characteristic is independent of the exact data type used to represent that attribute, so it does not matter if a number is represented as an integer, a real or a double precision float. Characteristics are specializations of the concept of object attributes that are further specialized in the Data Type concept, so each data type can have a specific set of properties. For example, complex numbers can have a method to calculate their real part. Images and procedures (methods) are two other attribute characteristics of this system, enabling various image representation formats (image data types) and image processing algorithms to be associated to objects like any other numerical or textual attributes.

To support the definition of images in SIRIUS, the following concepts are needed [Santos-97]: **Image Constants**, **Image Summarizers**, and **Image Parameters**. *Image Constants* are predefined images, meaningful in the application domain. For instance, a physician could have obtained a tomography where a given organ pathology appears in a predefined manner that could be used as a reference to search for or to identify other images with similar visual information. This image is defined as an Image Constant, so that it will be used by the system for purposes of standard comparison.

An *Image summarizer* is a procedure that operates over two images, extracting useful data from the image, called *Image Parameters*. An Image Parameters can be a numerical value, an image coordinate, a set of values, etc. Each Image Summarizer can return one or more Image Parameters, each representing some useful information for the process of image selection and retrieval.

Another construction was included to yield the many variations in which a target object can appear in an image. This construction is called a **modifier** whose purpose is to allow alterations on the default action of each image summarizer. The acknowledged variations in the images do not affect the basic behaviour of the summarizer, yielding image variations such as geometric transformations, contrast, etc. The application of any of these modifiers is directed at two categories of constant image information that must be retrieved: i) *Partial Constant Image*, where the comparison of one target image with the image constant can be found in sub-images of the image, and ii) *Full Constant Image*, in which each image constant must be entirely compared with the target image.

Using these concepts, images can be stored in association with objects in the database, and are treated as a characteristic of object attributes of a given data type. However, to speed up the queries involving image content data, the operational environment must be configured for the specific application domain. This is achieved by defining the set of image constants and image summarizers that are meaningful for each application. The idea is that, when an image is to be stored in the database, it is submitted to a predefined set of <image summarizer, image constants> pairs, and the resultant image parameters are used to create index structures, representing some form of "distance" of each stored image with this Image Constant.

When an application is built, the set of Image Constants related to each summarizer is defined, and the object attribute type summarizer needs to be evaluated. Thus, when each image is stored in the database as an attribute value, only the pairs <summarizer, constant> related to this particular association of the image are evaluated. Each of the resulting parameters is used to create the index structures, one for each image summarizer applied over each image constant meaningful to this object attribute. Each index structure is a modified R-tree [Guttman-84], representing the image in a N-ary space, where N is the number of dimensions of the particular parameter. In the retrieval environment, each parameter is attached to a user-defined range, so each query retrieves the images that each parameter has within the corresponding range. Each image can have indexes in many trees, one for each parameter of each summarizer meaningful for the object attribute type from which the image is a value.

In medical applications, it is very important to work with adjustable matching methods because images do not need to match an image constant exactly to be in the result set of a specific query. For instance, images that match 85% or more of the image constant can be considered a good answer, and they must be included in the index structure. Methods have been implemented that permit retrieval of images that match exactly with a stored image, and methods where similarity accepted as a match can be trimmed.

3. The Image-Contents Schema

The image support system has a meta-schema that represents the information contained in an object attribute of the image characteristic. The application domain is defined using objects of the type Image Constant, each having the attributes: *name*; *type* (partial or full Image Constant); and others describing the image. The characteristics that can be extracted from each

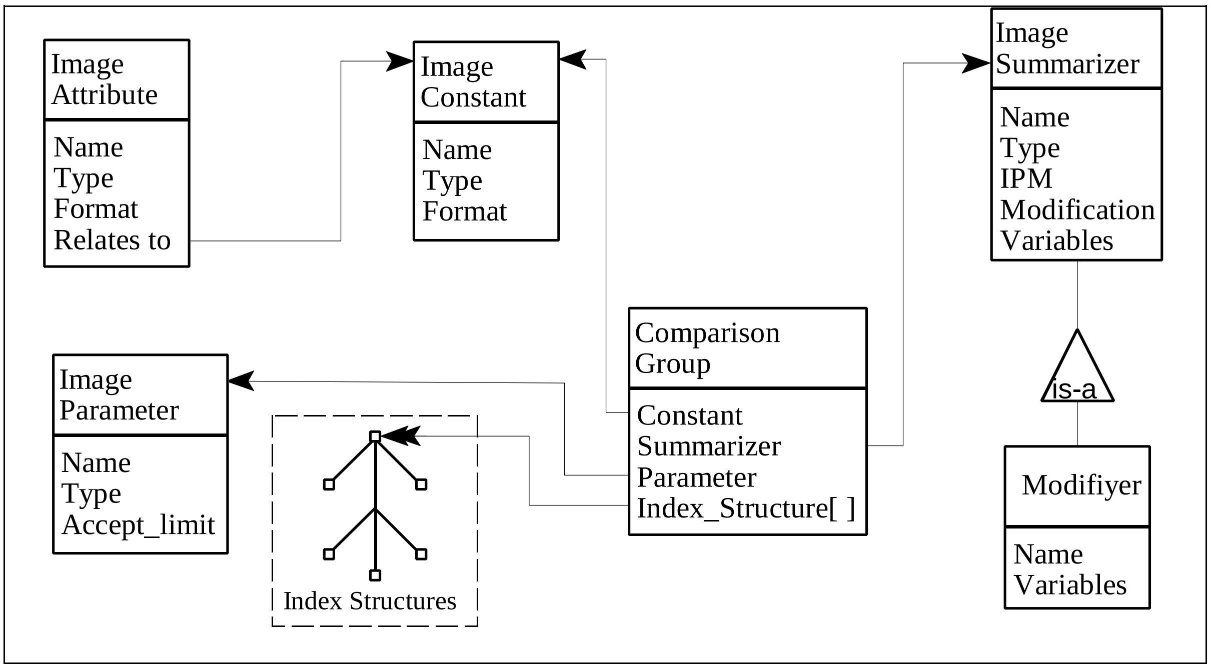


Figure 1: Meta-schema of an image supporting application**.**

stored image, through a comparison with one Image Constant, are represented as an object of the summarizer type, whose attributes are: *name*; *type* - meaning the image processing algorithm (histogram evaluation, matching searches, texture identification, etc.); and *modification* - an indication that the summarizer can have another definition for its default action. Each parameter returned by an Image Summarizer is represented as an object of the Image Parameter type. Its attributes are: *name*; *type* of information returned by the Summarizer; and an *Acceptance_Limit* defining the threshold from which an image will have one entry into the index structure. It also has one or more Image Processing Method - IPM, which is an Image Processing Algorithm implemented in some programming language and stored in the database. The Summarizer Modifiers are specializations (subtypes) of Summarizers, and thus are, in fact, alternative image processing algorithms used in particular situations, depending either on the query or on the object to which the image is attributed.

Figure 1 shows an overall diagram of the Meta-schema of this structure. The whole information extracted from one image when it is stored in the data base is represented by objects of type “Comparison Group”, relating objects of the Image Constant, Summarizer and Image Parameter types. This object type provides a semantic meaning for each value stored in the index structures. The index structures are reached from its instances (as represented in figure 1 by the small “tree” associated to this object type). The image itself is represented as an Attribute of the characteristic image, whose own set of attributes consists of: *Attribute_type_Name*, the name given to this type of attribute in the application schema; *Instance_Name*, an identifier of each image instance in the data base. Other attributes are specific for each of its instances, such as *length*, the *image* itself (the pixel matrix), *dimX*, *dimY*, and its *format*.

When an application is developed, its schema is added to the Meta-schema through the definition of some attributes of the data type “image”. Each attribute of the data type image, assignable to object types, must have at least one “image constant”, thus creating the link between the application schema and the meta-schema. For example, if objects of the user-defined type Patient have an attribute “lung tomography” defined in the application schema, then each object of the system-controlled type “Image Constant” whose values are interesting images of lung tomographies will provide the index structures to all images that are values of the attribute “lung tomography” of all objects of type Patient stored in the database.

4. A Content-Based Image Query System

This section describes how to build the modelling of the image attributes of an application, using a software tool developed to support the concepts presented herein. Such concepts enable: the definition of an application schema that stores images as attribute values associated to objects; the storage of images in the database and the query specification and retrieval of images from the database. The user interface of this tool is wholly based on forms and menus, although only a few of them are presented here. This tool was developed as a generic utility to support the manipulation of images in an application-independent way. For this reason, it is almost entirely dedicated to manipulate the schema of the applications, and has only a few resources supporting application-dependant queries.

This tool is, in fact, a graphical interface for an interpreter, which condenses the user action on the forms into a textual language called the Image Query Language (IQL). It is a dialect of SQL, extended with commands that permit queries to define and retrieve images using the presented concepts [Santos-97].

4.1. User Categories

The system described in this work presupposes the existence of three user categories: the developers, the production group and the clients. Developers are people responsible for the analysis and implementation of the application software inside a given application domain. They also develop the image processing algorithms, defining the IPM and their associated parameters.

The production group consists of the users that will configure the application and the image database, trimming them for use by the clients. They will define the Image Constants and the image processing methods meaningful for use as a summarizer for each object attribute of a data type image defined in the application schema. This group is subdivided into: a) the Image Database Manager - the person who works in collaboration with the developers and the medical staff to define the application schema; and b) the Application Managers, who work in collaboration with the Image Database Manager to define image retrieval and manipulation plans through predefined macros, to meet the needs of predefined queries already known.

The Clients are the physicians and end users, the people who use the application and the image retrieval and manipulation plans to query the system. By allowing the queries to be executed through predefined macros, the application manager can define the index structures that must be maintained by the Image Database Manager who, in turn, establishes what summarizers the developers must supply.

When a user sends queries to the database through the application, the predefined macros select a set of prospective images, narrowing the set of images that must effectively be processed, thus drastically reducing the processing time needed for the query. This technique induces the construction of expansible systems, where new Image Processing Algorithms can be dynamically appended into an application by the Image Database Manager, without code modifications. This can be done by linking the new Algorithm with its meaningful image constants through the attribute types. The results from the algorithm execution create one new index structure for each of its parameters, whose traversing can be embedded in the macros generated by the Application Manager. Figure 2 shows the architecture of the whole system, and the point where each user interacts with it.

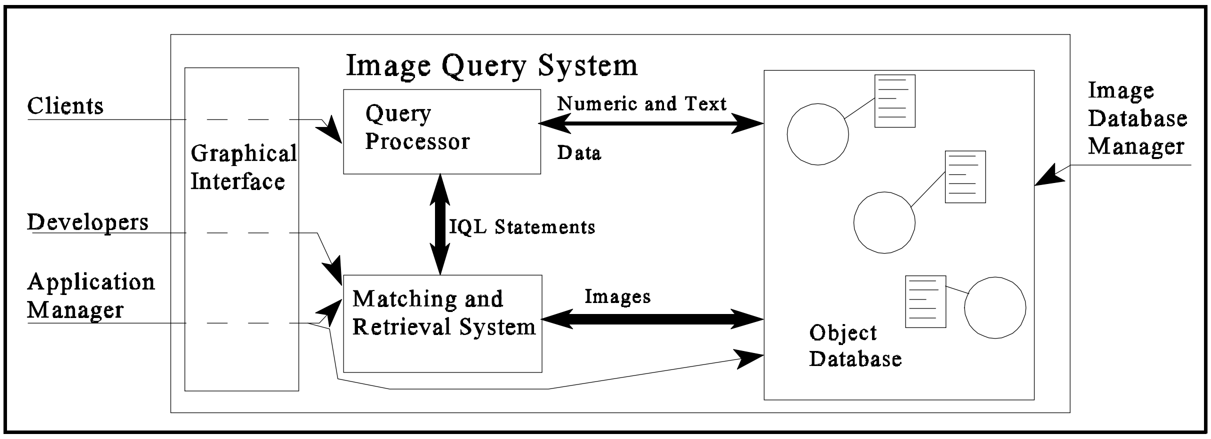


Figure 2: Architecture of the whole Image Content-based Retrieval System.

Whenever the system is started, the user must identify itself to the system through the usual ways, giving its name and password. The system recovers the user category and uses it to set the configuration of the main menu bar. Figure 3 shows the interface configuration presented to developers and users of the production group, who all have access rights to the whole system functionality. The interface hides the options to define the schema objects from the clients. These options are identified in the menu by the SchemaObject and Relationship items. The following sections present the options that developers and users of the production group can use to define the application schema, whose forms permit one to create, modify and delete elements of the schema.

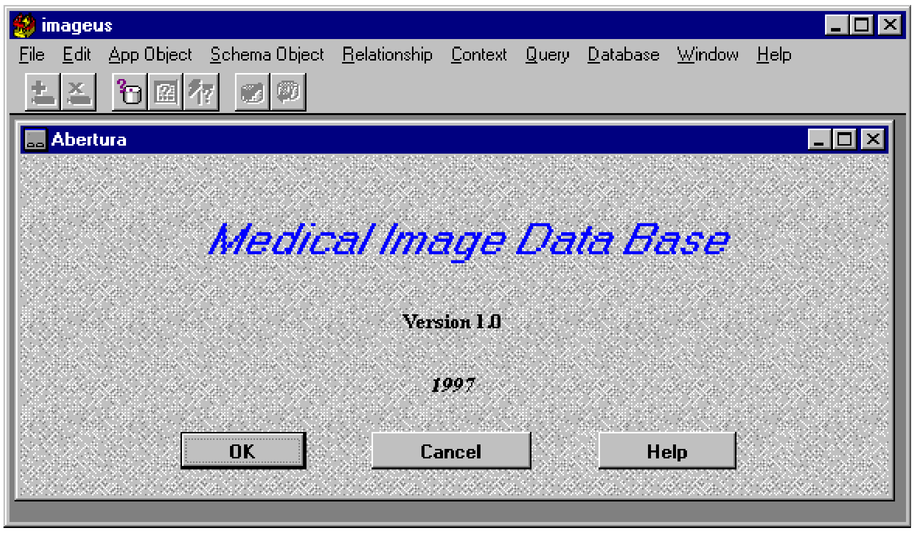

imageus
File Edit App Object Schema Object Relationship Context Query Database Window Help
Abertura
Medical Image Data Base
Version 1.0
1997
OK
Cancel
Help

Figure 3 - Opening window to developers and production group users.

4.2 - Defining Attributes of Image Characteristics

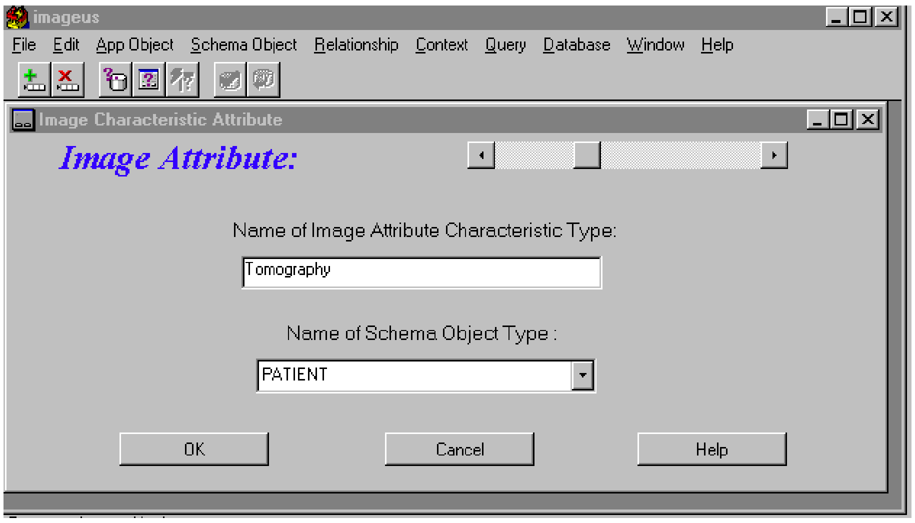
The association of attributes to objects and to relationships is usually made through the definition of the attributes directly in the command that declares the objects and relationship types. Using an IQL, the command has the following syntax:

```
      Define object <object_type> as
      (<attribute_name> [<attribute_characteristic>] <data_type> [<constraints>], ...);
```

For example, to define the object `Patient` as having the attributes `Name`, as a `text` of type `varchar`, `age` as a `number` of type `integer`, and `Tomography` as an `image` of the file format `.PCX`, the following command could be issued:

```
      Define object Patient as
            Name varchar(30) not null,
            Age Number Int,
            Tomography image PCX);
```

Using the Content-Based Image Query tool, this command is translated to a graphical interface, which performs the same function. Figure 4 shows an example of the form used to associate the attribute Tomography to the object type Patient. This form is reached through the option ImageAttribute of the SchemaObject menu item.


imageus
File Edit App Object Schema Object Relationship Context Query Database Window Help
Image Characteristic Attribute
Image Attribute:
OK
Cancel
Help

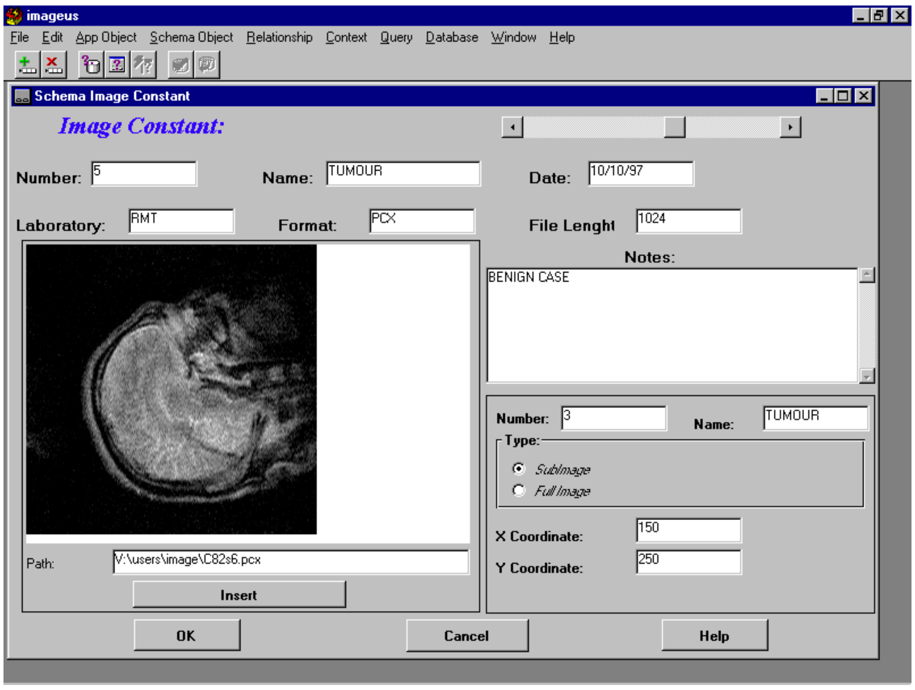
Name of Image Attribute Characteristic Type: Tomography

Name of Schema Object Type : PATIENT

Figure 4 - Association of an Image Attribute to an object type.

4.2 - Defining Image Constants

The first step to define image indexes is the creation of image constants. This is done with the Define Constant Image Form, accessed from the ImageConstant option of the SchemaObject menu item, as illustrated in figure 5. To define an Image Constant, its `Name` must be given, so it can be used in subsequent statements. The item `Type` defines if a constant image can be part of others. When a sub-image is selected, the items `coordinates X` and `coordinates Y` are used to express where inside the given image is a 32x32 pixels image. The clause Path is a general way to indicate where the image is. It can point either to an external file or to an already stored image. This form also permits the user to describe the image through additional attributes such as `Notes, Data, Laboratory,` etc.


imageus
File Edit App Object Schema Object Relationship Context Query Database Window Help
Schema Image Constant
Image Constant:
Insert
OK
Cancel
Help

Number: 5 Name: TUMOUR Date: 10/10/97

Laboratory: RMT Format: PCX File Lenght 1024

Notes: BENIGN CASE

Path: V:\users\image\C82s6.pcx

Number: 3 Name: TUMOUR

Type: [✓ SubImage] Full Image

X Coordinate: 150

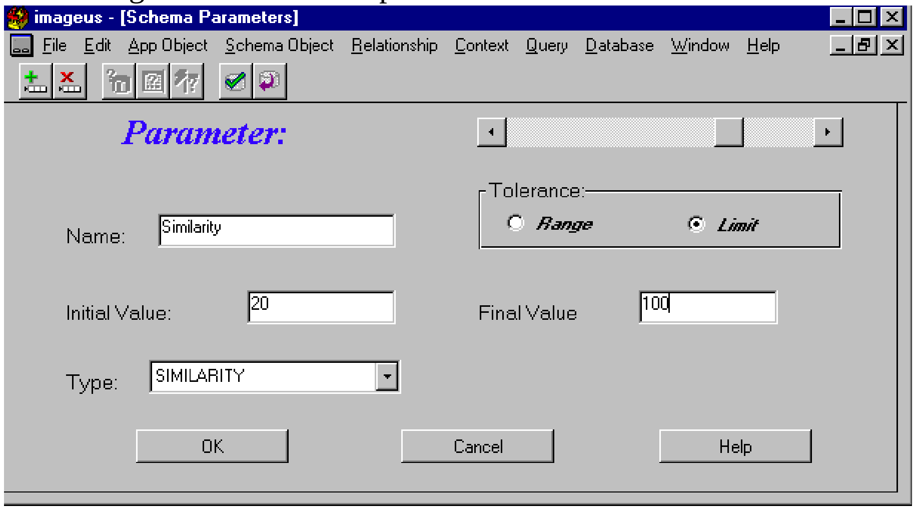
Y Coordinate: 250

Figure 5 - Defining an Image Constant.

4.2 - Defining image parameters

The specification of an image summarizer must represent each image parameter returned when it is executed over an image and an image constant. A summarizer is a conceptual element, which is implemented as an Image Processing Method written in some programming languages. The implementation of a method has a set of Internal Variables, which receives or returns values. In the same way that an IPM is the materialization of the Summarizer concept, the Internal Variables are the materialization of the Image Parameter concept.

The association of each image summarizer with the image parameter that it extracts is described later. The definition of an image parameter uses the option `Parameter` of the `Schema-Object` menu item, as illustrated in figure 6.


imageus - [Schema Parameters]
File Edit App Object Schema Object Relationship Context Query Database Window Help
OK
Cancel
Help

Parameter:

Name: Similarity

Tolerance: Range [✓ Limit]

Initial Value: 20

Final Value: 100

Type: SIMILARITY

Figure 6 - Assigning Image parameters to IPMs.

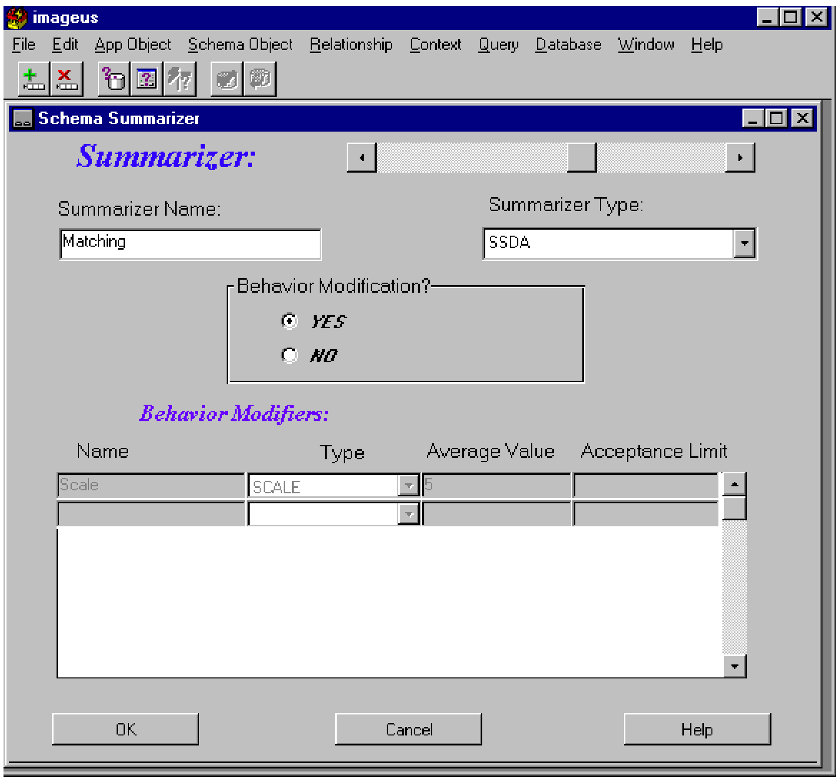
An image parameter consists of the definition of an identifier to reference it in the database, its association with an internal variable, and an acceptable tolerance.

The `Range` or `Limit` options give the tolerance threshold needed to put an entry of each image into the access structure of that parameter of a summarizer. The example in figure 6 shows the assignment of the internal variable SIMILARITY to the Image parameter Similarity, and the tolerance indicated by a limit from 50 up to 100 percent of similitude.

4.3 - Defining the image features to be extracted

a) Defining image processing methods as Image Summarizers.

To use image summarizers in an application, the corresponding IPM must have been previously developed in a programming language. Thus, it is necessary to assign each IPM to an identifier, so it can be reached from the tool as a summarizer. This can be achieved selecting the option `Summarizers` from the `SchemaObject` menu item, as shown in figure 7.


imageus
File Edit App Object Schema Object Relationship Context Query Database Window Help
Schema Summarizer
OK
Cancel
Help

Summarizer:

Summarizer Name: Matching

Summarizer Type: SSDA

Behavior Modification? [✓ YES] NO

Behavior Modifiers:

Name	Type	Average Value	Acceptance Limit
Scale	SCALE	5	

Figure 7 - Defining Image Summarizers and Modifiers.

Some IPMs have already been written in C++ language and are available to the applications (including the tool described here) through a library in DDL format. The following are the implemented IPMs to be used as image summarizers:

a) HISTOGRAM: maps the density distribution of the image to a defined number of levels;

b) SSDA: Searches for a specific pattern inside another image, using the Sequential Similarity Detection Algorithm [Brow-92].

c) MPPM: also an algorithm for pattern matching, using the Moment-Preserving Pattern Matching algorithm [Chou-90].

d) CORRELATION: a pattern matching algorithm based on the coefficient of the cross-correlation of two images [Gonzales-87] [Brow-92]. The tests performed to date showed that it is the matching algorithm best suited to be used with medical images.

Figure 7 illustrates the form `Summarizers` being used to create the Summarizer `Matching` associated with an already defined IPM called `SSDA`.

Each summarizer must specify the values of all of the input internal variables, and must validate the values received from the output internal variables. One IPM can be used to define more than one Summarizer, using some of its input internal variables to control its behaviour. It is assumed that any IPM must be constructed following a standard, which imposes a default value to be assigned to all of its input internal variables. If a summarizer needs to use the full power of an IPM, all of its image variables must accept this default value, and cannot be given a value. If an IPM is used in this way, other summarizers can be defined as Modifiers, specifying a set of Image Parameters for each that needs to receive specific values, such as an `Average value` and an `Acceptance limit` in the item `Behaviour Modifiers`.

The `SSDA, MPPM and CORRELATION` IPMs were implemented with the Internal Variable `DECIMAL THRESHOLD`, that sets the minimum coefficient between two images that can be considered a match. If not specified, this variable assumes the default value 0.5. The `SSDA` IPM was implemented with the additional Internal Variable `DECIMAL SCALE`, which allows the constant image to appear in the compared image on a different scale represented by an absolute value. If not specified, this variable assumes the default value one.

The `SSDA` IPM produces the following results:

- `DECIMAL CORRELATION`: the best correlation between the two images;
- `INT COORDINATEX`: the x coordinate in pixels of the best correlation obtained;
- `INT COORDINATEY`: the y coordinate in pixels of the best correlation obtained;
- `DECIMAL SCALE`: the Scale where the best correlation between the images was obtained.

The `MPPM` IPM produces the following results:

- `DECIMAL CORRELATION`: the best correlation between the two images;
- `INT COORDINATEX`: the x coordinate in pixels of the best correlation obtained;
- `INT COORDINATEY`: the y coordinate in pixels of the best correlation obtained;

The `CORRELATION` IPM produces the following results:

- `DECIMAL CORRELATION`: a coefficient list of the similar images matching two images;
- `INT COORDINATEX`: a list of the x coordinate in pixels of each similar image obtained;
- `INT COORDINATEY`: a list of the y coordinate in pixels of each similar image obtained;

The Histogram IPM has only the Internal Variable INT LEVEL implemented. It is used to establish the number of levels used to calculate the histogram - if not specified, it assumes the number 256. This IPM gives the following results:

DECIMAL MINIMUN: The minimum grey level obtained;
DECIMAL MAXIMUN: The maximum grey level obtained;
DECIMAL AVERAGE: The average level obtained;
DECIMAL DEVIATION: The standard deviation level obtained;
INT HLEVEL[ ]: an array of the grey levels obtained.

4.5 - Establishing relationships

Each stored image does not need to be compared with the full set of image constants. The definition of what pair <summarizer, image constant> should be applied over each incoming image is declared in two phases. The first corresponds to the specification of objects of the type “Comparison Group” shown in the meta-schema of figure 1. This is accomplished using the `Image Constant Description` Form, accessed from the `Relationships` Menu item, as illustrated in figure 8.

This form illustrates the definition of a comparison group specifying that an object named `TUMOUR`, of type Image Constant, must be compared with each image to be stored, using the image summarizer identified by `Matching`, and these comparison results are compiled in the parameters list containing `Similarity`, `X1_Position`, `X2_Position` and `PointsNumber`..

It is possible to define different comparison groups for the same summarizer and obtain different results. This provides flexibility to the queries, in the sense that only the comparison aspects required in a query need to be described. However, to be useful, a meaningful projection of the intended queries as part of the database project must also be produced. For example, the same summarizer Matching could describe other comparison groups, using other image constants (for example, one called heart) using the same or a different characteristic set.

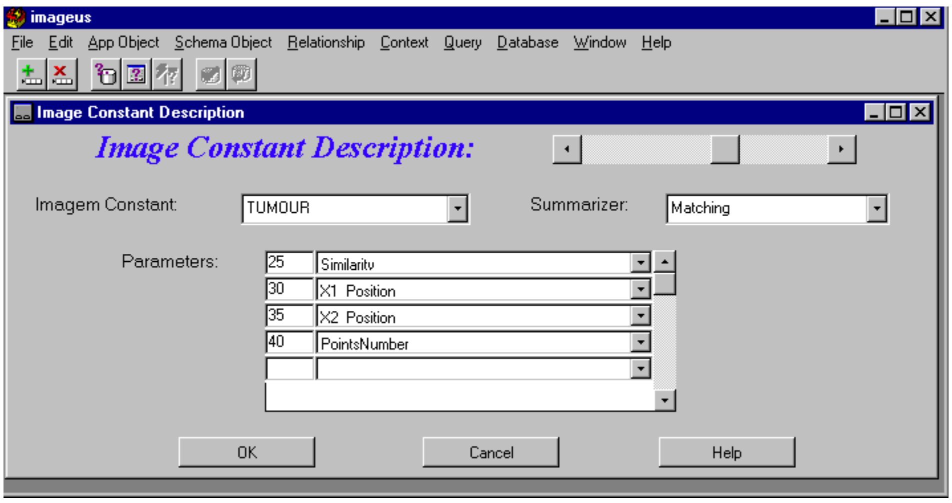

imageus
File Edit App Object Schema Object Relationship Context Query Database Window Help
Image Constant Description
Image Constant Description:
OK
Cancel
Help

Imagem Constant: TUMOUR
Summarizer: Matching
Parameters:

25	Similarity
30	X1 Position
35	X2 Position
40	PointsNumber

Figure 8 - Image Constant Description.

4.6 - Linking Object Attributes to Image Characteristics

The second phase to define what pair <summarizer, image constant> should be applied over each incoming image is performed when an object type is created. For each attribute of the characteristic image, a link is made to all the comparison groups of interest to this attribute. These links, which correspond to the "Relates to" attribute of the "Image Attributes" objects in figure 1, are established using the form illustrated in figure 9. This example declares that all images stored as values of attributes `Tomography` of objects of any type must be compared with the `Tumour` image constant.

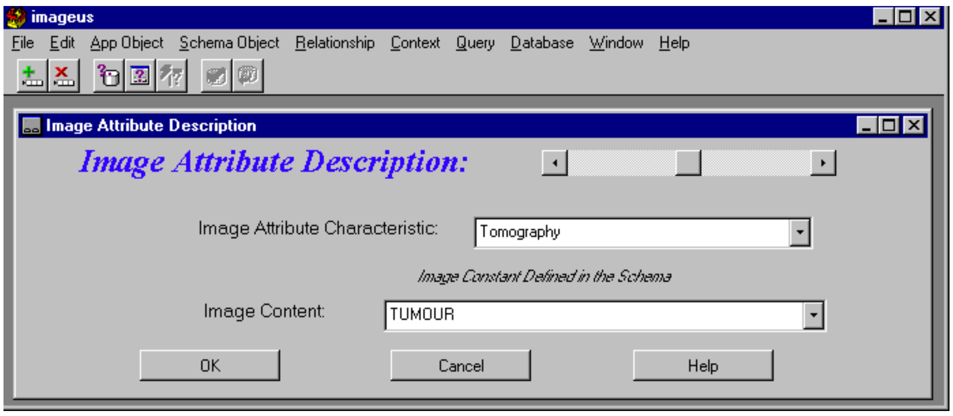


Image Attribute Description:

Image Attribute Characteristic: Tomography

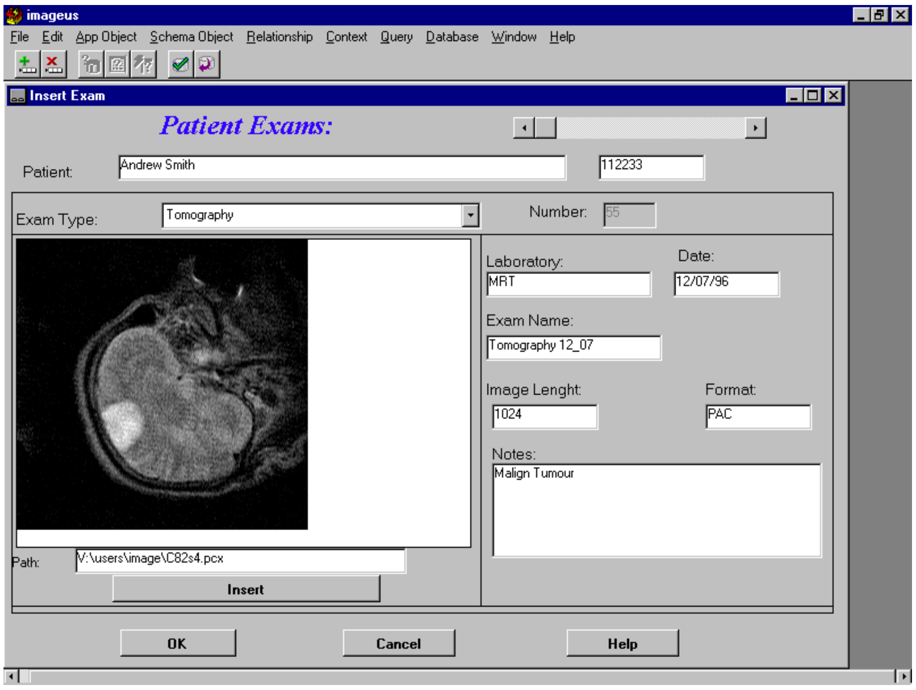
Image Constant Defined in the Schema

Image Content: TUMOUR

Figure 9 - Association of image attributes to Image Constants.

4.7 - Insertion of images as attribute values

Images can be assigned as values to attributes associated to already existing objects. Usually, this is an action taken by the user and is specific to each application. The Content-Based Image Query tool is application-independent and, in this respect, it does not manipulate the user-defined attributes specific to each user-defined object types. However, having been developed to be used to store medical images, a set of attributes commonly used in computer-based medical environments was defined. Therefore, this tool recognizes the name of the exam that corresponds to this image, the date when the image was taken, and the laboratory where it was taken. Assigning textual notes to each one is also possible. This information and the assignment of each image as an attribute value associated to an object uses the form Insert Image illustrated in figure 10. In this example, the image shown is being assigned as the value of the attribute Tomography, associated to the object of type Patient called Andrew Smith.



Patient Exams:

Patient: Andrew Smith 112233

Exam Type: Tomography

Number: 55

Laboratory: MRT

Date: 12/07/96

Exam Name: Tomography 12_07

Image Lenght: 1024

Format: PAC

Notes: Malign Tumour

Path: V:\users\image\C82s4.pcx

Figure 10 - Storing a image in the database.

4.7 - Image query commands

Queries are made using the form illustrated in figure 11. This form permits the definition of one comparison predicate, based on one attribute of the image characteristic . The example illustrated in figure 11 defines a predicate searching all stored images that are values of attributes Tomography, searched by the Matching Image Summarizer for the Image Constant Tumour, and having the Image Parameter Point-Number restricted by the acceptance limits.

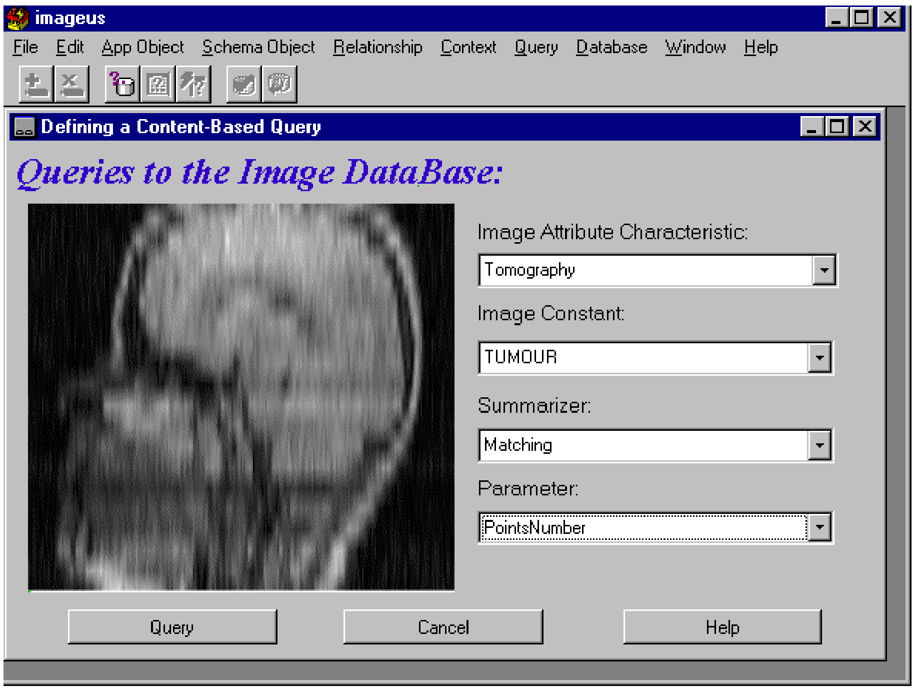

imageus
File Edit App Object Schema Object Relationship Context Query Database Window Help
Defining a Content-Based Query
Queries to the Image DataBase:
Query
Cancel
Help

Image Attribute Characteristic: Tomography

Image Constant: TUMOUR

Summarizer: Matching

Parameter: PointsNumber

Figure 11 - Defining a comparison predicate based on an image attribute.

The processing to recognize this image constant in all stored images of the Tomography

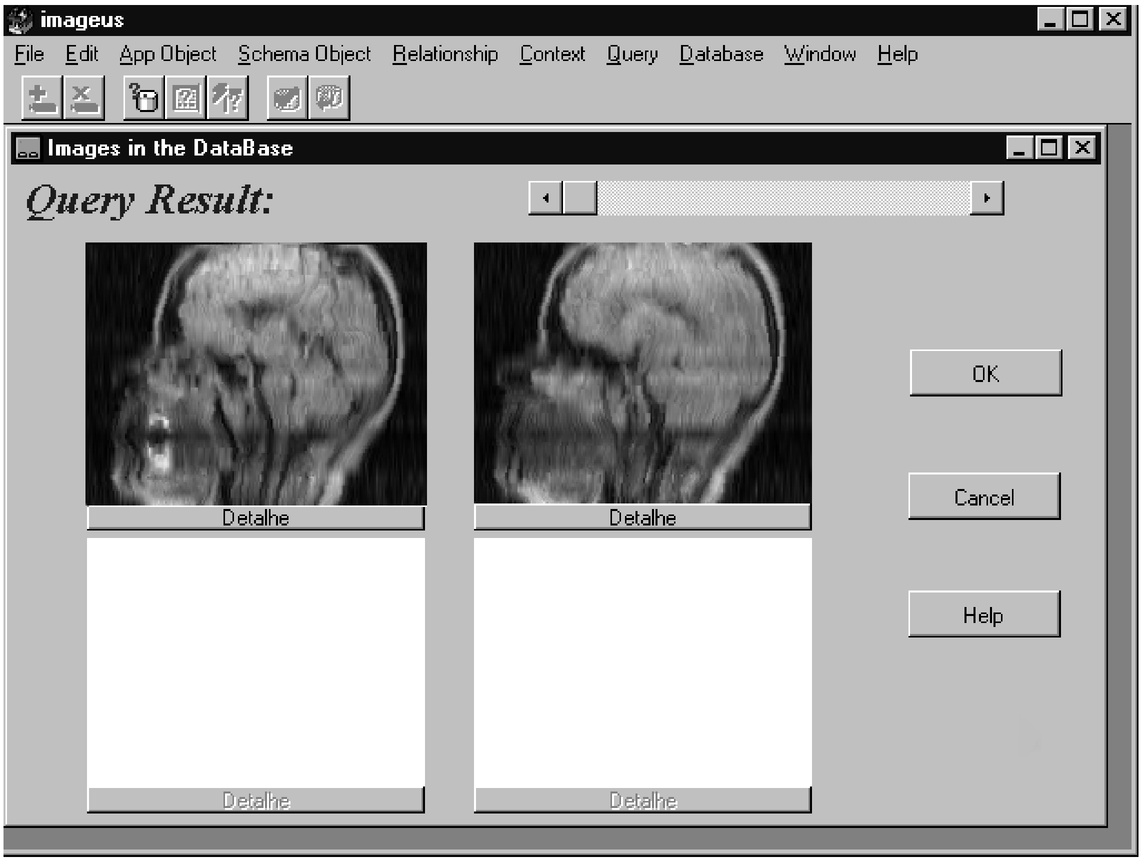

imageus
File Edit App Object Schema Object Relationship Context Query Database Window Help
Images in the DataBase
Query Result:
Detalhe
Detalhe
Detalhe
Detalhe
OK
Cancel
Help

Figure 12 - Result of a query using image attributes.

attributes was actually done when each image was stored. However, when the query is issued, no further image processing is executed: a fast search through an index structure selects the target images. Figure 12 shows the results of a query based on this single comparison predicate.

We have used this tool to measure the insertion and search times of images in a small database, containing approximately 50 images in each attribute. The time required to search in the database for queries involving up to five comparison predicates over one attribute is practically impossible to measure because it is so short. The retrieval of each image identified in the response set takes up to 0.3 seconds using a Pentium 133MHz with 64MBytes of main memory, running version 7.3 Oracle server in the NT 4.0 environment. This server stores the image schema definition and the application data (including the images). Much of the processing time is used to store an image in the database. This time is related to the number of Image Constants associated to the image attribute to which the image is assigned and to the number of comparison groups involving that Image Constant. However, this time is almost independent of the number of images already stored in the database. In a typical situation, when using the cross-correlation IPM and the Histogram IPM with four image constants, it takes about 8 minutes to store each image in the database. It must be noted that if the described concepts are not used, the storage time of each image is not significant. However, each query issued takes about 2 minutes to process each stored image by each comparison predicate, regardless of the number of images retrieved in the response set. Thus, a query issued against the same database (50 images) involving 5 comparison predicates takes about 6:35 hours.

5. Conclusion

This work describes the use of the concept of images as a data type and Image Constants and Image Summarizers as a way to pre-process images and speed up the content-based retrieval of images stored in an Image Database. These concepts are presented using a software tool to exemplify the definition of a schema of how images are related to an application schema. The concepts presented herein can be used in cooperation with more traditional searches based on descriptive text strings and icons attached to the images, although the described method can almost always replace the others. The proposed approach maintains the precision of the search by storing the entire image.

Image retrieval processing is accelerated by bringing the workload of the image processing methods to where the images are stored. This improves the use of computer-processing time because usually the insertion of images in the database occurs distributed in time, as each is collected during the daily operation of a medical laboratory or health centre. It also avoids wasting computer power because the results of the execution of the Image Processing Methods over an image are always stored, so each computation is done only once.

This approach can use a set of indexes to quickly discard many images that undoubtedly are not part of the response set of a query, enabling submission of only a reduced set of images to the more time-consuming filters needed to complete the answer. A new set of commands to enable the definition of such structures was added to an object-oriented version of the Structured Query Language (SQL).

The fact that modifications in an image by a given image processing algorithm correspond to equivalent changes in the image parameters returned by an image summarizer permits manipulation of the index structures accordingly. Thus, the image modifiers establish operations over the images homomorphic to operations over the index structures. However, the operations over the index structures are much less time-consuming, so when queries require operations over image constants to be used in the image comparisons, those operations over images can be mapped to operations over the index structures. This leads to a new range of optimizations that can be explored in the image content-based queries, to be researched in future.

6. References

[Biajiz-96] Biajiz, Mauro - “**Modelling Data Models Using Abstraction Parametrizations**” (in portuguese), *PhD Thesis* presented to IFSC - University of Sao Paulo - Brazil, September 8th, 1996.

[Brown-92] L.G. Brown, “**A Survey of Image Registration Techniques**”, *ACM Computing Surveys*, Vol. 24, No. 4, pp. 325-376, Dec 1992.

[Cardenas-93] A. F. Cardenas at all, “**The knowledge-based Object Oriented PICQUERY+ Language**”, *IEEE Trans. on Knowledge and Data Engineering,* Vol. 5, No. 4, pp. 644-656, Aug. 1993.

[Chang-92] S. Chang and A. Hsu, "**Image Information Systems: Where do we go from Here?**", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 4, No. 5, Oct 1992, pp. 431-442.

[Chow-90] C.H. Chow and X.C. Chen, “**Moment-Preserving Pattern Matching**”, *Pattern Recognition*, Vol. 23, No. 5, Oct 1990, pp. 461-474.

[Dutton-97] G.F. Dutton, “**The Current State of PACS**”, *Applied Radiology,* Aug. 1990, pp 15-19.

[Gonzales-87] R. C. Gonzales; P. Wintz - “**Digital image Processing**”, Addison Wesley, 2nd Edition, 1987.

[Gudivada-95] V.N. Gudivada, V.V. Raghavan, “**Content-Based Image Retrieval Systems**”, *IEEE Computer,* Vol. 28 , No. 9, Sep. 1995, pp 18-22.

[Guttman-84] A. Guttman, “**New Features for Relational Database Systems to Support CAD Applications**”, *PhD Theses,* University of California , Berkeley, Jun. 1984.

[Santos-97] R. R. Santos - “**Fitting Image as a new Data Type in an Object Oriented Database Management System**” (in portuguese), *PhD Thesis* presented to IFSC - University of Sao Paulo - Brazil, 1997.

[Senzako-97] E. Y. Senzako, **“SisMatch - Matching for Magnetic Resonance Imaging**”, (in portuguese) *MS Theses,* University of São Paulo Oct 1996.

[Tannús-94] A. Tannús, A. Torre Neto, M. J. Martins, T. J. Bonagamba, N. Beckman, J.F.W. Slaets, H. Panepucci, “**Architecture of 2.0 Tesla NMR Tomograph**”, in *Proc. of the 9th International Conf. of the Society of Magnetic Resonance,* Aug 6-12, 1994, S. Francisco, USA, vol III, pp. 1101-1109.

[Traina-97] Traina Jr, C.; Traina, A. J. M.; Santos, R. R.; Senzako, E. Y. - "**Content-Based Medical Image Retrieval in Object Oriented Databases**", in *Proceedings of the 10th IEEE Symposium on Computer-Based Medical Systems,* Part II, pp. 67-72, June 1997, Maribor-Slovene.

[Vidoto-86] E.L. Vidoto, H. Panepucci, A. Tannús, M.J. Martins, **“A New Type of Head Coil for MRI in Ultra Low Magnetic Field**”, in *Proc. of the 2nd meeting of the Society of Magnetic Resonance*, June 20 - July 06, 1986.