

# Design and Implementation of a content-based Image Retrieval Tool

Giancarlo Iannizzotto<sup>1</sup> Antonio Puliafito<sup>2</sup> Lorenzo Vita<sup>1</sup>

<sup>1</sup> Dipartimento di Matematica, University of Messina  
C.da Papardo - Salita Sperone  
98166 Messina - Italy

E-mail: ianni@scirocco.unime.it, vita@mat520.unime.it

<sup>2</sup> Istituto di Informatica, University of Catania  
Viale A. Doria 6, 95025 Catania - Italy  
E-mail: ap@iit.unict.it

## Abstract

In this paper we describe the design and implementation of the Java Automated Image Storage and Retrieval (*JAISR*) tool, a recording and retrieval system for digital images, based on *Retrieval By Shape Similarity*. The system has been designed in order to permit the retrieval of the images recorded relying only on an approximate sketch of one of the objects contained in the image searched. It is equipped with a user-friendly graphic interface, can be accessed through the network in a client-server mode, and is totally platform-independent, since it requires only the presence of a Java-enabled browser on the client.

**Keywords:** Image indexing and retrieval, Java, Turning angles, Security

## 1 Introduction

During the last few years, a considerable development in the area of multimedia technology has been noticed. The natural consequence of such development has been a great interest for archivation systems for non-traditional data and informations, especially digital images and videos. In the past years, the problem of archiving such non-traditional data was solved by extending conventional

databases, in order to allow them to deal with the so-called *BLOBs*, Binary Large Objects, that is, something similar to "black boxes" containing data whose form and type was not of interest for the archiving system, whose only function was to put data in the file system. The task of indexing the data was thus given to human operators who had to characterize images with alphanumeric strings, generally linked to the content of images themselves; such alphanumeric strings were registered in the database, and only those informations were actually managed by it. The queries used to be made on strings, and their results consisted of one or more records, each containing some information fields and a path within the file system; such path used to lead to the file containing the digital image, in the form of a Blob. Even if such approach can be very useful in several occasions, present technology may provide us new consultation paradigms for a digital image repository, in which the queries made by the user are not based on a sequence of strings conventionally chosen for representing the elements of the record, but on information *similar to the recorded data* (in our case, therefore, on graphic information). Besides, it would often be convenient to manage *uncertain queries*, through which we could search not just a specific object, but all those objects similar to it, according to specific similarity queries [15] [6] [20].

Even if, in the past few years, the management of uncertain queries has been the object of research and application development in text Information Retrieval and the results often enabled the development of commercial applications, the same thing cannot be easily said about the area of research about Content-Based Image Retrieval. In text information retrieval, the approach most commonly used for searching for a document somehow related with a specific subject or having some features (author, title, language, structure, etc.) consists of characterizing each document in the record, during the insertion, according to specific parameters, and using such parameters in another moment as keys for research. The difference between such techniques and the ones that we described before (which are based on a human operator) is given by the fact that in the last case the elements of the record are characterized *in a totally automatic way*, that is, with no human intervention. This causes, as well as a speeding up of operations, a higher efficiency of the system, since the keys for the research are automatically fixed according to characteristics of the element to be recorded, and not according to the operator's opinions. However, in the most part of the cases, the *features* used for characterizing the documents have the same level of abstraction as the data contained in the documents themselves: they are alphanumeric strings. We can significantly remark that, in a system that can manage content-based image retrieval, the abstraction level of queries made by the user is much higher than the data contained in a common digitized image, which normally consists of a matrix of values, each representing the color or the gray level of a point of the image (PIXEL: PICTURE Element). They are based on information concerning the shape of the objects contained in the image, their color, the texture of surfaces, or on their position in the space of the image [7][3][21]. In this case, the level of

the *features* which we need to use for the retrieval must therefore be high; this causes other problems: for example, in the case of Retrieval By Shape Similarity, we need to find a suitable technique for representing forms, and an adequate measure of similarity among them, so that we can compare each other the objects contained in the images, by using quite their similarity.

Whichever is the set of features selected for representing the entities of the archive, it is anyway convenient to define a *distance* among the elements on the domain of such set, so that we can effectively distinguish them and we can, above all, determine a level of similarity among them. This is necessary for satisfying the already mentioned *uncertain queries* . In the next paragraphs, we will broadly describe a recording and retrieval system for digital images, based on *Retrieval By Shape Similarity*, designed in order to permit the retrieval of the images recorded relying only on an approximate sketch of one of the objects contained in the image searched. Such system was equipped with a user-friendly graphic interface, can be accessed through the network in a client-server mode, and is totally platform-independent, since it requires only the presence of a Java-enabled browser on the client.

## 2 Shape-Based Image Retrieval

The retrieval by shape similarity is one of the most interesting areas in the research on multimedia database. There are two reasons for such interest. On one hand, the possibility of searching for an image starting from a sketch or an example, is indeed one of the most important features for a user of the system. On the other hand, giving such opportunity involves very serious problems, that today have not fully been solved, yet. According to what has been pointed out by researchers in the computer vision field, such problems are mainly due to two factors [6] [5]:

- the representation of the shape of the objects, which must be compact in order not to make the phase of research too heavy , but, at the same time, meaningful enough to give an adequate accuracy of results;
- the comparison among shapes, because similarity among objects very often greatly depends on the context in which the application operates. For example, in some cases, invariance with respect to rotations, shifting, scaling of the objects contained in images may be desirable; in other cases, it may not.

Several methods of representation and comparison among shapes have been proposed throughout the years. Some of them are based on combinations of area, perimeter, factors of shape, barycentre, main axes and/or of symmetry of the objects and a set of invariable algebraic moments [9] [26] [28]. Other methods are

based on measures which are considered to be closer to human visual perception, as a function of curvature and *Turning Angles* [25] [1], which should enable the user of the system to measure similarity more satisfactorily. On the other hand, such methods are generally harder, from the computational point of view and regarding the resources required. The use of adequate indexing techniques is therefore required, in order to obtain acceptable performances.

In J-AISR we choose the latter solution: the image undergo a series of elaborations (described in [14] [13] [8]) from which a feature vector for each object in the image is extracted; this feature vector is a Turning Angle representation of the outline of the object, i.e. it represents its *shape*.

### 3 Indexing Issues

Today many experts think that the most important requirement for an image database is an effective and quick storage, search and retrieval capacity, within very large digital image repositories. In order to reduce average and worst case time required for query operations, we therefore need to develop very effective indexing techniques, which could enable the system to grow in size without excessive decrease in performances and accuracy, and which could also support similarity queries: such techniques are called *similarity indexing* [29] [21].

Similarity indexing is used for meeting three requirements:

- high dimensionality of feature vectors representing the images or the objects they contain is a problem for the indexing policies that are commonly used today, because they cannot generally deal with more than two or three dimensions, without causing serious problems for effectiveness;
- the ability to satisfy similarity queries usually requires a domain expert to find out one or more distances that could measure the similarity (or dissimilarity) between two feature vectors;
- the particularity of some types of queries.

The indexing policies and the corresponding data structures that have to meet such requirements are often very different from those used for traditional data. The solution generally adopted is to deal with each feature vector as a point in a  $n$ -dimensional space, and to use a multidimensional indexing method. Some interesting multidimensional indexing methods are known as Spatial Access Methods (SAM). They are data structures specifically devised for managing large collections of  $n$ -dimensional points stored to disks, in order to make researches quick and effective. During the last few years, several SAM have been developed: an interesting survey can be found in [24]. The ones that in the past were considered more promising were the extensions to research trees (k-d tree

[4], k-d-B tree [22], hB-tree [19], cell-tree [10], R-tree [11]) and, among them, R-tree, in their several variations proposed (R-tree, packed R-tree [18], R+-tree, R\*-tree). A type of R-tree has recently been proposed, in which factual technology is applied [17]. All those multidimensional indexing methods seem to suffer from a significant decrease in performances when the number  $n$  of dimensions increases, so that they become useless when  $n$  exceeds some tens. The most recent approaches to this problem follow two ways: the first one consists of developing new data structures and indexing techniques that could scale well according to the dimensionality of feature vectors and, at the same time, to the number of items of the database (such as TV-tree [16] and the new SS-tree [30]). The other, more conservative way consists of mapping higher dimensional data to a lower dimensional space before indexing [2], [6]. The advantage given by this second option is that we can continue to use tools that have already been widely tested, during the last few years, in several applications, such as R-tree and R\*-tree. While waiting for researchers in this area to come out with a new best-performer, multidimensional well-scaling similarity indexing method, we decided to use the easiest way. We exploited a new method to reduce the dimensionality of feature vectors, and used as database engine one of the best and most widely used tools: Postgres. Postgres is a complete and well-working DBMS, it is object-oriented, can be extended through functions given by the user, and supports dynamic libraries and *large objects*. Moreover, it has a strong support for managing R-trees. In next paragraph we will consider the procedure that we used for reducing the dimensionality of feature vectors.

## 4 Making things look closer

For reducing the dimensionality of feature vectors, we need to project such vectors into a space that has a lower number of dimensions; in general, such projection implies a limited loss in information, that may be accepted if some basic conditions (linked to the type of queries that we are going to deal with) are not violated. For example, let us suppose that the user can produce only queries of this type:

*Find all the objects **identical** to the object  $X$ ;*

the only condition needed is that, if two objects  $X$  and  $Y$  exist, such as their feature vectors  $A$  and  $B$  in the original space are the same, then their equivalents in the reduced dimensionality space will be the same, too.

That is:

$$A = B \implies A' = B' \tag{1}$$

where  $A'$  and  $B'$  are respectively the feature vectors of  $X$  and  $Y$ , projected into the reduced dimensionality space.

Such condition, even if it is always necessary, is not sufficient if we also need the possibility of managing similarity queries, such as nearest-neighbour queries:

*Find the  $k$  objects **more similar** to the object  $X$ ;*

or range queries:

*Find all the objects which are within a similarity degree  $\epsilon$  to the object  $X$ ;*

in this case, two other properties must be added to the first one.

Let  $X, Y$  and  $Z$  be three objects, and let  $A, B$  and  $C$  be respectively their feature vectors; then, let  $A', B'$  and  $C'$  be the corresponding feature vectors in the reduced dimensionality space; finally, let  $D()$  be the distance fixed on the original space of  $A, B, C$ , and  $D'()$  the distance fixed on the reduced dimensionality space. Thus, if we want to permit the nearest- neighbour query, we need to prove the property:

$$D(A, B) \leq D(A, C) \implies D'(A', B') \leq D'(A', C') \quad (2)$$

Instead, in order to permit range queries, we need to verify the property:

$$D(A, B) \leq \epsilon \implies D'(A', B') \leq \epsilon \quad (3)$$

where  $\epsilon$  is a positive non-null quantity.

This property corresponds to the fact that, due to the projection, objects that in the original space would have been chosen as meeting the requirements of the query, do not risk to be considered not valid (*no false dismissals*).

On the other hand, the properties that we have thus formulated do not protect us in any way from *false hits*, that is, the cases when some objects that in the original space would be discarded, are considered valid. In general, a small percentage of false hits is acceptable, as a price to be paid for the certain advantages which derive from the reduction of dimensionality.

In practice, instead of satisfying the property that we have just enunciated, we can assure another one which implies it. If we keep the previous notation, we can summarize it as follows:

$$D'(A', B') \leq D(A, B) \quad \forall A, B \quad (4)$$

The equation that we have just proposed is like to say that, according to the distance fixed in the dimensionally reduced space, objects result to be closer to each other than they would be according to the distance fixed in the original space. In other words, in the projection space, *things look closer* [].

## 5 The adopted indexing solution

Postgres supports R-tree indexing for three dimensions: our projection space needs therefore have three dimensions at most. Since feature vectors, in our case, represent bidimensional outlines, our simplest idea was to sort such vectors according to their similarity to one or more reference shapes. After several

tests and heuristic considerations, we chose only two reference shapes: circle and triangle. In fact, if we suppose that we always deal with closed outlines, the representations in Turning Angles for such two shapes are different enough to consider them orthogonal (see figure). Of course, such options are not strict, and they may therefore be adjusted. In any case, a different option in reference forms would not affect the solutions adopted.

Each feature vector becomes therefore a point of a bidimensional space in our system, and the co-ordinates of such point are respectively the distance (in the original space) of the feature vector from the reference circle and the distance from the reference triangle. Expressing it in a formula, if a feature vector  $P$  has the co-ordinates  $(X_P, Y_P)$  in the projection space, and another feature vector  $Q$  has the co-ordinates  $(X_Q, Y_Q)$ , if we call  $D(\cdot)$  the distance in the original space, and respectively  $R_1, R_2$  the two reference feature vectors (the Turning Angles of the circle and of the triangle), we will obtain:

$$\begin{cases} X_P = D(P, R_1) \\ Y_P = D(P, R_2) \end{cases} \quad (5)$$

and:

$$\begin{cases} X_Q = D(Q, R_1) \\ Y_Q = D(Q, R_2) \end{cases} \quad (6)$$

We only need to define an appropriate distance on the projection space. For the sake of convenience, we can define (on the projection space) distance  $D'(\cdot)$  as a means of the common Euclidean distance, in order to keep its *metric* properties in the space where it is defined. Expressing it in a formula:

$$D'(P, Q) = \frac{1}{2} \sqrt{(X_P - X_Q)^2 + (Y_P - Y_Q)^2} \quad (7)$$

We can immediately prove that (4) is valid also for the distance that we have just defined:

**Property 1** *The defined distance  $D'(\cdot)$  is always less-than or equal-to the primitive distance  $D(\cdot)$ :*

$$D'(P, Q) \leq D(P, Q), \forall (P, Q) \quad (8)$$

*Proof:*

$$\begin{aligned} D'(P, Q) &= \frac{1}{2} \sqrt{(X_P - X_Q)^2 + (Y_P - Y_Q)^2} = \\ &= \frac{1}{2} \sqrt{[D(P, R_1) - D(Q, R_1)]^2 + [D(P, R_2) - D(Q, R_2)]^2} \leq \\ &\leq \frac{1}{2} \sqrt{(|D(P, R_1) - D(Q, R_1)| + |D(P, R_2) - D(Q, R_2)|)^2} \end{aligned} \quad (9)$$

and therefore

$$D'(P, Q) \leq \frac{1}{2} (|D(P, R_1) - D(Q, R_1)| + |D(P, R_2) - D(Q, R_2)|) \quad (10)$$

if we apply the triangular inequality to both the terms that are in the second member, and if we remember that the commutative property is valid for metric  $D(\cdot)$ , we obtain:

$$\begin{aligned} |D(P, R_1) - D(Q, R_1)| &\leq D(P, Q); \\ |D(P, R_2) - D(Q, R_2)| &\leq D(P, Q) \end{aligned} \quad (11)$$

from which the proof comes.

□

What we have just proved enables us to apply the distance that we have defined on the projection space, in order to operate range queries, knowing that the use of such distance will cause false alarms but not false dismissals. If false alarms are not acceptable, since their number is generally low, they can always be eliminated with another operation, by applying the original distance in the original space to the elements resulting from the first selection. The computational overload due to this second operation will always be low, if compared with what we save by reducing the dimensionality. Filtering techniques like this are sometimes called *quick and dirty filtering*, and have already been used in the past, for example in QBIC Project of IBM [2].

## 6 System Architecture

The indexing technique described was tested in an image registration system able to support a content-based image retrieval based on the shape of the objects contained in images themselves, called J-AISR. J-AISR is a client-server and distributed extension of AISR system, which we have developed during our research at the University of Messina [14][8]. It can therefore be accessed, through the network, by any computer with a Java-enabled browser. The structure of J-AISR server is modular, and consists of three basic blocks in cascade:

1. Java superserver: it is a daemon written in Java language, waiting for the clients' connection requests, and functions as an interface with the rest of the system;
2. AISR is the image analysis system: it segments the images, isolating the objects that they contain, and extracts the feature vectors that have to be stored and used for indexing. This block is completely written in Ansi C language;

3. the real archive. It is implemented with Postgres, enriched with functions related with distances  $D(.)$  and  $D'(.)$ , written in Ansi C.

The previous versions of AISR, even if they were based on the same image analysis and shape representation technologies as J-AISR, did not have specific forms of indexing and were stand-alone, not supporting remote access. J-AISR was developed making wide use of Java technology to make it accessible from any node connected with the Internet, as long as it possesses a Java-enabled Web browser. To make the tool available only to authorized users, adequate security mechanisms based on public and private key algorithms are included, which provide authentication services and, if required, encryption. The approach proposed was successfully used to port the Sharpe tool (Symbolic Hierarchical Automated Reliability/Performance Evaluator) onto the Web [23].

## 6.1 Communication Mechanisms

The approach we followed can be seen as an extension of the client/server programming paradigm.

Unlike the classical approach, however, the client does not need to possess any specific software; through a simple Web interface it loads the software using the mechanisms provided by Java. The immediate advantage is thus the simplicity of access to the application and the total absence of a preliminary phase to distribute and install the interface software. The application provider can then update the application, modifying the interface as he likes, without having to provide potential clients with updated versions of the software. As the proposed approach entrusts the server with execution of the application, the task of correctly sizing it belongs to the service provider. Moreover, it is possible to detach the work performed on the client from the calculations made by the server. And all this is possible by moving around on a Web page which is easy to use and does not require an interactive session to be held for any length of time.

The only requirement for the client is a Java-enabled Web browser, while the server needs the following software modules:

- Web server;
- Java Virtual Machine;
- Application to be made available on the network;
- Java applet of the user interface;
- Software module to run the communication session with the client.

The last module, entirely developed in Java, in reality comprises two sub-modules. One of these in particular is transferred onto the client when the latter forwards a request for access to the server and provides the client with the mechanisms needed to run the communication sessions just started. The second submodule, on the other hand, is always in execution on the server and deals with accepting requests from various clients, robustly managing the various connections with clients, and sending clients the results put out by the server. It also keeps a memory of the correspondence between clients and the applications they use.

To describe the functioning of the technique proposed in greater detail, we refer to Fig. 1, which is a scheme of the various phases of interaction between client and server.

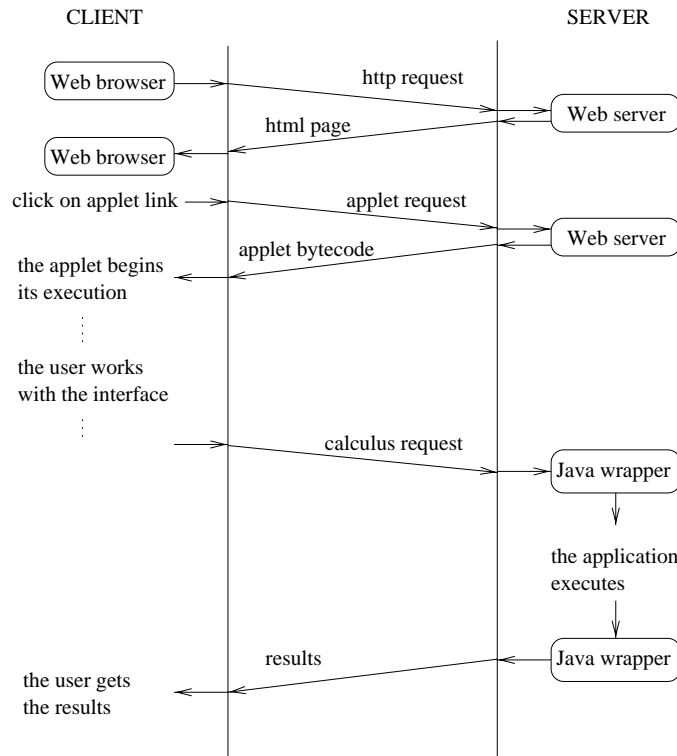


Figure 1: Interaction between client and server

Through his Web browser, with a simple click of the mouse, the client sends a request in http format to the server, who transmits the html page in question. On this page there will be a link providing access to the application which allows a Java applet to be loaded, containing the graphic interface developed for the software involved. At this point the first phase of the client-server interaction ends. The client now has the software to view and run the graphic interface with which to formulate the request to be sent to the server. The user can thus specify his request using his own computing resources. When this phase ends the request

is transmitted to the server, possibly after conversion into a format accepted by the specific software, which is then executed on the data provided by the user. The entire request for computing is now shifted to the server, while the client waits for the processing results. At the end of this phase the server sends the client the processed data.

The presence of security mechanisms, if any, involves an extension of the first phase of the communication protocol to check the real identity of the user. Further details of how to implement a security service will be given in the following section.

## 6.2 Security and Access Control

Network sharing of J-AISR tool requires the creation of suitable security mechanisms to regulate access, reserving access to previously authorized users. It is therefore necessary to provide authentication services and, if required, encryption, which will be all the more necessary if the applications put into the network are of the commercial kind. The techniques we will use are based on public and private key algorithms [12, 27].

Below we will refer to a server S and a generic client C that wants to use the services provided by S. In our structure the server S also represents the authority issuing certificates for the clients' public keys. The communication protocol can be subdivided into the following 3 stages, shown in Fig. 2: *registration*, *initialization* and *data transfer*.

**Registration Stage:** In this stage of the security protocol C and S agree on the type of services offered/required and the access modes. C then generates two keys, a private one and a public one, giving the latter to S through a safe channel (typically not through the same communication channel that subsequent connections will use) and keeping the private one safely (the key is often encrypted with an alphanumeric password and then memorized). To S, receipt of C's public key represents a request for registration. S then creates a certificate for C's public key, signing it with its own private key, after which it delivers its own public key to C (again on a safe channel) along with the certificate for C's key. It should be pointed out that this stage is only carried out once in the validity period of the registered key.

**Initialization Stage:** This stage starts when C decides to link up with S to use a certain application. Neglecting the loading of the applet through Web, as any security mechanisms are incorporated in the Web browser, this stage can be schematically represented as follows:

- C sends a connection request to S;
- S sends a signed message indicating that the request has been received;

- C replies sending a signed message containing the security services required (confidentiality and authentication or authentication alone), and a code identifying its public key;
- S checks C's signature on the message received and if it is recognized, sends C a signed acknowledgement and starts to transfer the data; if the authentication stage fails the connection is immediately interrupted.

**Data Transfer Stage:** during this stage each message sent is treated according to the security services requested. If confidentiality and authentication have been requested the message will be composed of two fields, one in which the data is encrypted and another containing the sender's signature.

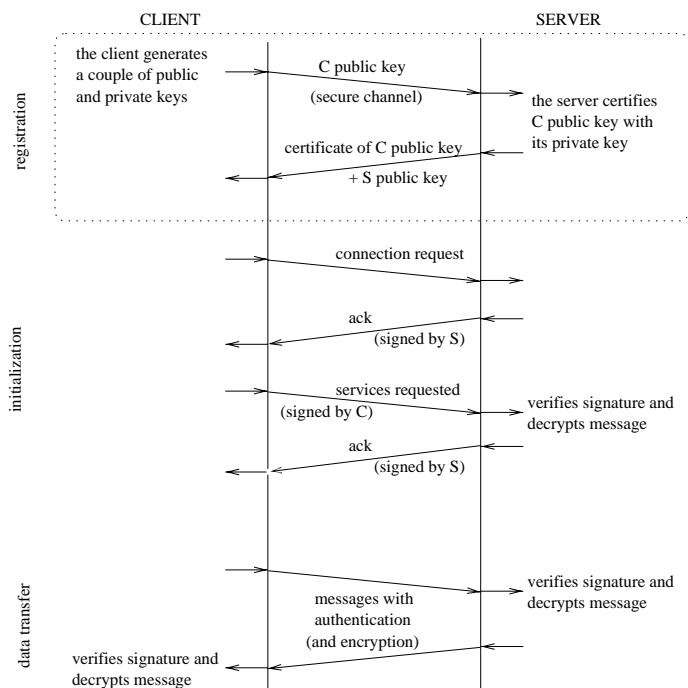


Figure 2: Security protocol

## 7 Application example

In this Section we present the Graphical User Interface of J-AISR and the results obtained from a generic user query.

Actually, the whole J-AISR server is installed on a Pentium 100 PC, with 4GB of hard disk and 16 MB of ram. The operating system is Linux, and the Postgres version is Postgres95 1.0.1, compiled with GNU gcc 7.2. Several shape distance function have been tested with J-AISR and AISR: in fact, the  $D'(\cdot)$

distance described in this work is compatible with any distance function  $D(\cdot)$  chosen in the original space, provided that it is a metric. In particular, we have tested the Euclidean distance and the Median distance [14] [8].

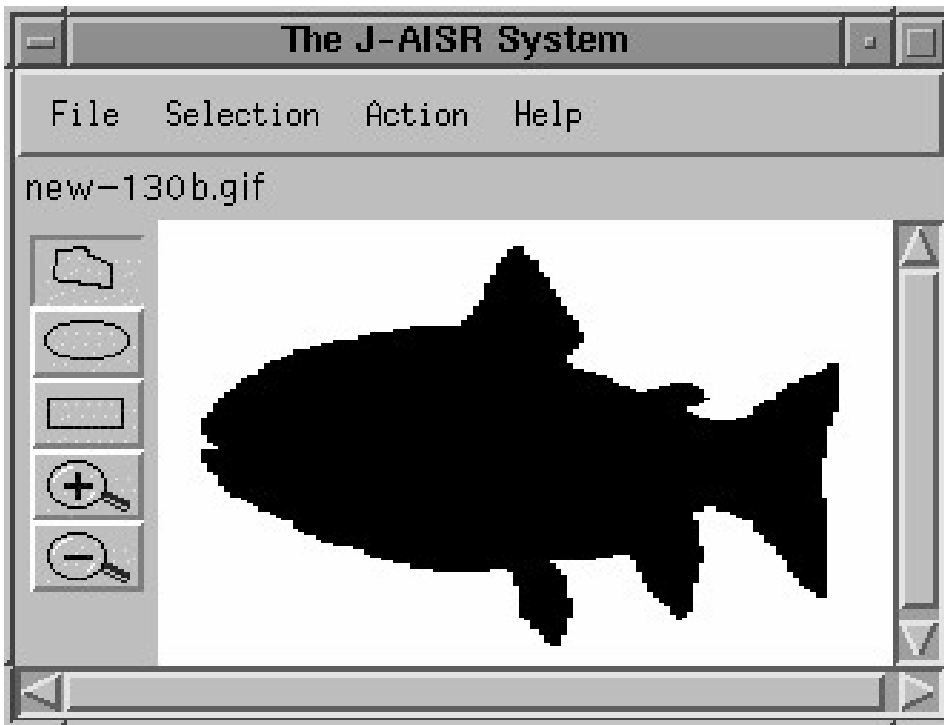


Figure 3: Main display of J-AISR

The main J-AISR display, shown in Fig. 3, has the following four zones: a Menu panel; a design area; a status panel.

Besides the usual choices, such as the submenus *File* and *Help*, the Menu offers the *Selection* submenu with items which allow the user to choose the way in which objects from the image can be selected. The following options are available: *free hand*, *ellipse* and *rectangle*. The former activates a pointer which allows the user to selected arbitrary shapes, while the latters simply allow to choose ellipse or rectangular selections of parts of the image. More immediate use of these functions is provided by a series of push buttons on the left hand side of the display which summarize the main options on the menu. The user is also given the possibility to enlarge and reduce the image he is working on. The submenu *Action* contains two items: *Search* to activate the search phase and *Option* which allow the user to specify all the parameters used in the searching phase.

The design area is where the image to search for is loaded or where the user can provide his graphical input for the search phase.

Finally, the status panel gives run-time indications regarding the status of the

interface, signalling the occurrence of any event that may be of interest to the user.

Once the input specification stage is completed, the user passes to the searching stage simply by pressing the *Search* key which activates the procedures described in the previous Sections for contour extraction and image retrieval.

At the end of the processing stage, the results are transmitted back to the client. A new window is then opened which shows a sequence of images (the number can be chosen by the user) sorted according to the distance value from the query originally submitted. Double clicking on one image produces a new window with an enlarged version of the image.

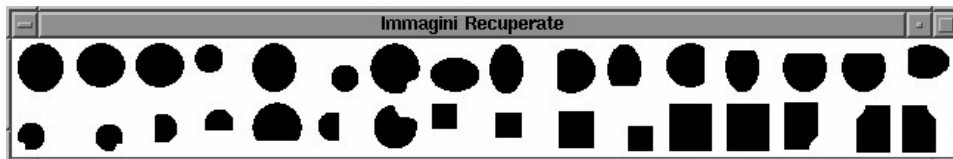


Figure 4: The results of a query: the searched image is the first, the images are sorted using the Euclidean Distance

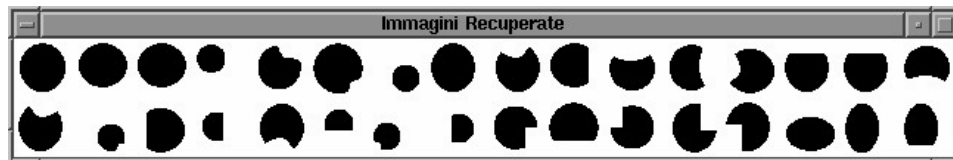


Figure 5: The results of a query: the searched image is the first, the images are sorted using the Median Distance

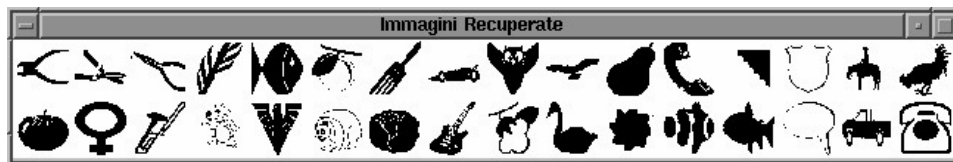


Figure 6: The results of another query: the searched image is the first, the images are sorted using the Median Distance

In the example shown in Figure 4, the user searched for an image which was present in the database, so this image is the first of the list; the images are sorted by using the Euclidean Distance.

In the example shown in Figure 5, the query is the same of the previous example, but the images are sorted according to the Median distance. Even though the results look very similar, the Median distance gives less *false hits*.

In the example shown in Figure 6, the user searched for the *pliers* shape: it should be noted that this is not a simple case for the system, but it correctly retrieved three pliers images. Again, the adopted distance was the Median one.

## 8 Conclusions

We have developed a complete and effective system for image retrieval by shape similarity. The system has been implemented using a client-server configuration, using Java language for the interface and the communication subsystem. The advantages of the proposed approach can be summarized as follows:

- the opportunity to access the database from any client, irrespective of the hardware architecture and operating system, as long as a Java-enabled Web browser is available;
- occupation of the network only when really necessary;
- elimination of problems relating to the installation, configuration and maintenance of software.

At this moment, J-AISR exploits Turning Angles as the representation technique for object shapes: for this technique we developed the necessary algorithms for calculating and comparing the feature vectors, and a novel indexing approach which reduces the retrieval time without losing in accuracy. The system performs well, showing a response time which scales well with the dimension of the database and which is on average of 35" for a database of about 500 images.

## References

- [1] E.M. Arkin, L.P. Chew, D.P.Huttenlocher, K.Kedem, and J.S.B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on PAMI*, 13(3), March 1991.
- [2] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic. Automatic and semi-automatic methods for image annotation and retrieval in qbic. Technical report, IBM Research Center of Almaden, 1995.
- [3] G. Phanendra Babu, Babu H. Metre, and M. S. Kankanhalli. Color indexing for efficient image retrieval. *Multimedia Tools and Applications, Kluwer Academic Publisher.*, 1, November 1995.
- [4] J.L. Bentley and J.H. Friedman. Data structures for range searching. *CM Computing Surveys*, 11(4), Dec. 1979.

- [5] Shi-Kuo Chang. *Principles of Pictorial Information Systems Design*. Prentice-Hall International Editions, 1989.
- [6] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3, 1994.
- [7] M. Flickner, H. Sawhney, J. Hafner, W. Niblack, J. Ashley, D. Petkovic, Q. Huang, B. Dom, M. Gorkani, D. Lee, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *Computer*, 28, September 1995.
- [8] A. Puliafito G. Iannizzotto and L. Vita. A new shape distance for content based image retrieval. In *Multimedia Modeling, MMM96*, Toulouse (France), November 1996.
- [9] Y. Gong, H. Zhang, H.C. Chuan, and M. Sakauchi. An image database system with content capturing and fast image indexing abilities. In *Proceedings of International Conference on Multimedia Computing and Systems*, Boston, May 1993.
- [10] O. Gunther. The cell tree: an index for geometric data. Technical Report UCB/ERL M86/89, Univ. of California, Berkeley, Dec. 1986.
- [11] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD*, June 1984.
- [12] L. Hughes. *Actually Useful Internet Security Techniques*. New Riders Publishing, 1995.
- [13] G. Iannizzotto and L. Vita. A fast, accurate method to segment and retrieve object contours in real images. In *International Conf. on Image Processing*, Lausanne, Switzerland, 1996.
- [14] G. Iannizzotto., L. Vita, and A. Puliafito. Aisr: an automated image storage and retrieval system. Technical report, University of Messina, 1996.
- [15] H.V. Jagadish. A retrieval technique for similar shapes. In *Proc. SIGMOD 91 Conf.*, Denver, 1991.
- [16] H. Jagadish K. I. Lin and C. Faloutsos. The tv-tree - an index structure for high-dimensional data. *VLDB Journal*, 3:517 – 542, Oct. 1994.
- [17] I. Kamel and C. Faloutsos. Hilbert r-tree: An improved r-tree using fractals. Technical report, University of Maryland CS Dept.
- [18] I. Kamel and C. Faloutsos. On packing r-trees. Technical report, University of Maryland CS Dept.

- [19] D.B. Lomet and B. Saltzberg. The hb-tree: a multiattribute indexing method with good guaranteed performance. *ACM TODS*, 15(4), Dec. 1990.
- [20] R. Mehrotra and J. E. Gary. Similar-shape retrieval in shape data management. *Computer*, 28, September 1995.
- [21] E.G.M. Petrakis and C. Faloutsos. Similarity searching in large image databases. Technical report, University of Maryland, 1995.
- [22] T. Robinson. The k-d-b-tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of ACM SIGMOD 81*, 1981.
- [23] R. A. Sahner, K. S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems*. Kluwer Academic Publishers, November 1995.
- [24] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989.
- [25] Brian Scassellati, Sophoclis Alexopoulos, and Myron Flickner. Retrieving images by 2d shape: A comparison of computation methods with human perceptual judgments. In *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 2185, pages 2–14, February 1994.
- [26] S. Sclaroff and A.P. Pentland. Modal matching for correspondence and recognition. *IEEE Trans. on PAMI*, 17(6), June 1995.
- [27] W. Stallng. *Network and Internetwork Security Principles and Practice*. Prentice Hall, 1995.
- [28] G. Taubin and D.B. Cooper. Recognition and positioning of rigid objects using algebraic moment invariants. In *Geometric Methods in Computer Vision, SPIE 91*, 1991.
- [29] D. A. White and R. Jain. Similarity indexing: Algorithms and performance. In *SPIE 96*, San Jose (CA), Feb. 1996.
- [30] D. A. White and R. Jain. Similarity indexing with the ss-tree. In *12<sup>th</sup> IEEE Intl. Conf. on Data Engineering*, New Orleans, Louisiana, Feb. 1996.