A Methodology for the Representation, Indexing, and Retrieval of Images by Content[1]

Euripides G.M. Petrakis and Stelios C. Orphanoudakis

Department of Computer Science, University of Crete
and
Institute of Computer Science, Foundation for Research and Technology-Hellas
Heraklion, Crete, Greece

Abstract

This paper considers the requirements for the design and implementation of an image database system which supports the storage and retrieval of images by content. Attention is focused on a specific methodology for the efficient representation, indexing, and retrieval of images based on spatial relationships and properties of objects. Images are first decomposed into groups of objects and are indexed by computing addresses to all such groups. This methodology supports the efficient processing of queries by image example and avoids exhaustive searching through the entire image database. The performance of an image database system using the above methodology has been evaluated based on simulated images, as well as images obtained with computed tomography and magnetic resonance imaging. The results of this evaluation are presented and discussed.

1 Introduction

In application domains such as remote sensing, astronomy, cartography, meteorology and medical imaging, images comprise the vast majority of acquired, processed and archived data. The medical imaging field, in particular, has grown substantially in recent years and has generated additional interest in methods and tools for the management, analysis, and communication of medical image data. Picture Archiving and Communication Systems (PACS) are currently used in many medical centers to manage the image data produced by computed tomography (CT), magnetic resonance (MRI), digital subtraction angiography (DSA), digital radiography, ultrasound, and other diagnostic imaging modalities which are available and routinely used to support clinical decision making. It is possible to extend the capabilities of diagnostic imaging modalities and provide added value to PACS by developing and image database (IDB) system which also supports the analysis, storage, and retrieval of images by content.

Important considerations in the design and implementation of IDB systems are: image feature extraction, image content representation and organization of stored information, search and retrieval strategies, and user interface design. Traditional database concepts such as data independence, data integrity, and control of shared information are also of great significance. Recent proposals regarding the design of IDB systems and the management of image data are influenced by the "object oriented" approach [1, 2]. This approach offers a framework

[1]Image and Vision Computing, No. 11, Vol. 8, pp. 504-521, October 1993.

within which different types of entities (e.g., different kinds of image data) and operations (e.g., image processing functions, image access mechanisms etc.) may be uniformly represented as “objects”. Objects are grouped into “classes” which can also be objects. Object classes are organized into hierarchies thus, taking advantage of the property of “inheritance”.

Image data can be distinguished into: original image files, called “physical images”, and image related data (i.e., descriptions, attributes etc.), called “logical images” [3]. Physical and logical images are stored into a physical and a logical database respectively. There may be a need to treat physical and logical data independently since, in most cases, they serve different purposes. Access methods are usually applied to logical data rather than the vast amount of physical data. Furthermore, logical images are subject to updates, while physical images are permanently stored (e.g., on an optical disc), and retrieved only for viewing.

The effectiveness of an image database system, which supports the archiving and retrieval of images by content, ultimately depends on the types and correctness of image representations used, the types of image queries allowed, and the efficiency of search and retrieval techniques implemented. In selecting an appropriate type of image representation, an attempt must be made to reduce the dependence on the application domain as much as possible and to ensure some tolerance to uncertainty with regard to image content. Furthermore, image representations must be compact to minimize storage space, while image analysis and search procedures must be computationally efficient in order to meet the efficiency requirements of many IDB applications.

The content of images can be determined, based on the correspondence between the derived description of a particular image and some appropriate model(s) of an image class [4, 5]. So far, most of the techniques which have been developed are knowledge-based, making use of application and domain-specific knowledge and are not particularly well suited for image retrieval by content and general IDB work; this is mostly due to problems related to computational efficiency, uncertainty, and general knowledge representation issues. To deal with with the problems of efficiency and uncertainty in deriving reliable image representations in IDB systems, one solution is to perform image processing and analysis interactively and under supervision.

The retrieval capabilities of an IDB must be embedded in its query language. Command oriented query languages allow the user to issue queries by conditional statements involving various image attributes (values of attributes and/or ranges of such values). Other types of image queries include: queries by identifier (a unique key is specified), region queries [6] (an image region is specified and all regions that intersect it are returned), text queries [7] etc. The highest complexity of image queries is encountered in queries by example. In this case, a sample image or sketch is provided and the system must analyze it, extract an appropriate description and representation of its content and, finally, match this representation against representations of images stored in the database. This type of query is easy to be expressed and formulated, since the user need not be familiar with the syntax of any special purpose image query language.

Query response times and the size of the answer set depend highly on query type, specificity, complexity, amount of on-line image analysis required and the size of the search. In addition, query formulation ought to be iterative and flexible, thus enabling a gradual resolution of user uncertainty. All images (and/or information related to images) satisfying the query selection criteria should be retrieved and displayed for viewing. Query responses can be further refined by “browsing”: before a final selection is made, characteristic representations (e.g., icons, image miniatures) corresponding to all images contained in the answer set are displayed. Browsing can be especially helpful when the specification of pictorial content is ambiguous and it may be the only method for making a difficult final selection [8]. In addition, displaying such image forms, instead of the original images, avoids extensive data transfers through a network during retrievals.

So far, in order to determine which images must be retrieved, content representations corresponding to all stored images are compared (one by one) with a similar representation extracted from the query image. Thus, retrievals can be inefficient due to the fact that comparisons often involve time intensive operations such as graph matching [9, 10]. Various other techniques, with lower time complexity, can be used to resolve such queries. Such a technique is matching based on "2-D strings" [11]. A successful match associates the query image with only part of a stored image (a subset of the objects it contains), but the whole image is retrieved.

In this paper, attention is focused on a specific methodology for the representation, indexing and retrieval of images by content. This methodology supports the efficient processing of queries by image example and avoids exhaustive searching through the entire IDB. Images may be indexed and accessed based on spatial relationships between objects, properties of individual objects, and properties of object classes. A given image is first decomposed into groups of objects, called "image subsets". All image subsets up to a predefined maximum size are considered. An image is then indexed based on addresses computed to all the derived image subsets.

This work completes and extends the work of others. In particular, the image representations used in this work may be considered as an extension of the representation of "2-D strings" [11]. Based on this representation, an indexing scheme and a retrieval strategy are proposed, which in contrast to 2-D strings, avoid exhaustive searching through the entire IDB. Moreover, previous work [12] on the indexing of images by content may be considered as a special case of the proposed indexing scheme and is based on appropriate encodings of all pairs of objects (i.e., image subsets of size 2) derived from all stored images.

The rest of this paper is organized as follows: the functional characteristics of a prototype IDB system for medical images, in which the methodology presented in the following sections will be integrated, are discussed in Section 2. The representation of image subsets is presented in Section 3. The indexing scheme and the storage file structure used are presented in Section 4. The retrieval of images by content, the search strategy, and the evaluation of the performance of retrievals are discussed in Section 5. The application of the proposed methodology to an image database consisting of medical images is presented in Section 6, followed by conclusions in Section 7.

2 IDB System Characteristics

A prototype medical IDB system is currently under development in our laboratory. The system is based on the integration of tools and methodologies which support the interactive processing, classification and browsing of medical images, as well as methodologies for the efficient automated indexing, storage and retrieval of such images by content. Earlier implementations of this system have been previously described in [13, 14, 15].

The IDB system under development is capable of handling various types of medical image data including, the raw images themselves, attributes (e.g., dates, names), text (e.g., diagnosis related text), information extracted from images by automated or computer assisted image analysis, modality and image file header information (e.g., ACR/NEMA [16]) etc. Image data are further distinguished into physical and logical.

Logical image data are organized into classes forming hierarchies. Different kinds of classes can be identified and used to assist application modeling and to reduce the search space in specific queries. Specifically, logical image data may be organized into classes based on anatomy, diagnosis, imaging modality etc. Knowledge in the form of procedures (e.g., image processing and retrieval procedures corresponding to a specific class), rules, and parameters may be assigned to each class and may be inherited by the lower level classes. In designing and developing an IDB system which satisfies the needs for a hierarchical database organization, takes advantage of the property of inheritance, and is extensible, the object-oriented approach has been adopted.

Physical data (images), are stored on a separate image store (e.g., on an optical disc) and organized in clusters based on the likelihood of being retrieved together in response to a particular query (e.g., the set of all images of the same patient may be stored in adjacent locations on the disc). Pointers are implemented from the logical to the physical database.

Commercial object oriented DBMSs continue to emerge, each providing different functionalities and obeying its own design principles. As a consequence, given the requirements of an application, the decision of which data model to use is not an easy one and, therefore, adopting one of the currently available models may not turn out to be proven to be the best solution in the long term. We have chosen to base the development of our prototype medical IDB system on a low-level storage subsystem (storage manager) [17, 18]. A storage subsystem provides mechanisms for the persistent storage and retrieval of objects. The subsystem to be selected must be extensible with DBMS features such as mechanisms for defining, creating, modifying and accessing both the data model (database schema) and the data. In addition, it must be extensible so that it can provide transaction management, concurrency control, authorization for a multiuser environment etc. A higher level subsystem is built on top of the storage manager and designed to support the desired functionality. The front-end module of this subsystem consists of an interactive user interface environment which supports the communication between the user and the various system components. In particular, this environment supports the following:

Image Processing. It includes tools and mechanisms for image filtering, image registration, and the interactive or automated segmentation of medical images (see Section 3).

Image Classification. Images may be classified into one or more predefined classes of an anatomical or a diagnostic hierarchy. An image may be simultaneously an instance of more than one classes found at any level in a particular hierarchy. The instances of each class may be stored (logically or physically) separately, so that the IDB is partitioned into database segments. This reduces the search space leading to faster retrievals, since a query addresses only a specific database segment.

Image Queries. Queries may be specified by: (a) identifier, in which case the value of a single key (e.g., an image file name) is specified, (b) conditional statement involving values (or ranges of values) of various attributes, (c) an example image (grey level image or sketch) and (d) a combination of the above. A query language integrating within its syntax all kinds of image queries is under development. A user may interactively specify the specific database segment to be searched; otherwise, the whole hierarchy will be searched. All images satisfying query selection criteria need to be retrieved, while a user is allowed to make a final selection by browsing. Images and image related data (e.g., text, attributes) stored in the database may both have to be retrieved in response to a particular query.

Image Browsing. The user interface is equipped with interactive graphical tools and mechanisms for both displaying and selecting image classes or class properties, and for exploring their interrelationships. Such display tools are referred to as “graph browsers” in [13]. Furthermore, the user interface is equipped with an “image browser” whose purpose is to display reduced resolution images (miniatures) or sketches corresponding to images stored in the database.

Once the content of images has been extracted reliably, the system resumes responsibility for the efficient automated archiving and retrieval of images. The effectiveness of the IDB system is significantly enhanced by incorporating into its storage and search mechanisms the methodology which is presented in detail in the following sections.

3 Image Representation

The search and retrieval of images by content, using the method to be described in this paper, relies on image descriptions based on the segmentation of images into dominant disjoint regions or objects. Although accurate and robust image segmentation techniques are currently becoming available, this is an independent area of active research and is beyond the scope of this paper. However, it should be pointed out that the requirement for accurate and robust image segmentation is more relaxed for indexing and retrieving images by content than it is for image analysis and image understanding. Thus, we have opted for a conventional image segmentation technique resulting in polygonal approximations of objects contours. The desired segmentation results are obtained by editing (i.e., the user may delete insignificant segments or correct the shape of others). The edited segmented forms are then used to compute a variety of image features constituting a particular image representation, and for efficient browsing of the query response sets. Figure 1 shows (a) an example of an original grey-level MRI image and (b) its corresponding final segmented polygonal form. The segmented image contains 6 objects numbered 0 through 5.

An image containing n objects is decomposed into groups of image subsets. The subsets in each group contain an equal number of objects k, with k ranging from 2 up to a prespecified number K_{max}. In particular, $k \in [2, \min(n, K_{max})]$. An image subset of size k can be viewed as an answer to a possible image query specifying k objects. Therefore, K_{max} can be set equal to the maximum number of objects allowed in queries, if such a value can be specified in advance. In general, the value of K_{max} depends on the application (see Section 5.4).

Producing all subsets of a given image containing n objects is equivalent to generating all “combinations” of n elements taking them k at a time. Taking all subsets for all values of k, results in a total of $\sum_{k=2}^{\min(n,K_{max})} \binom{n}{k}$ subsets. For example, the image of Figure 1 which contains 6 objects, gives rise to 15 image subsets of size 2, 20 subsets of size 3, 15 subsets of size 4, 6 subsets of size 5 and 1 subset of size 6 (which is the original image itself).

Prior to indexing, image subsets are ordered. Each object can then be associated, through

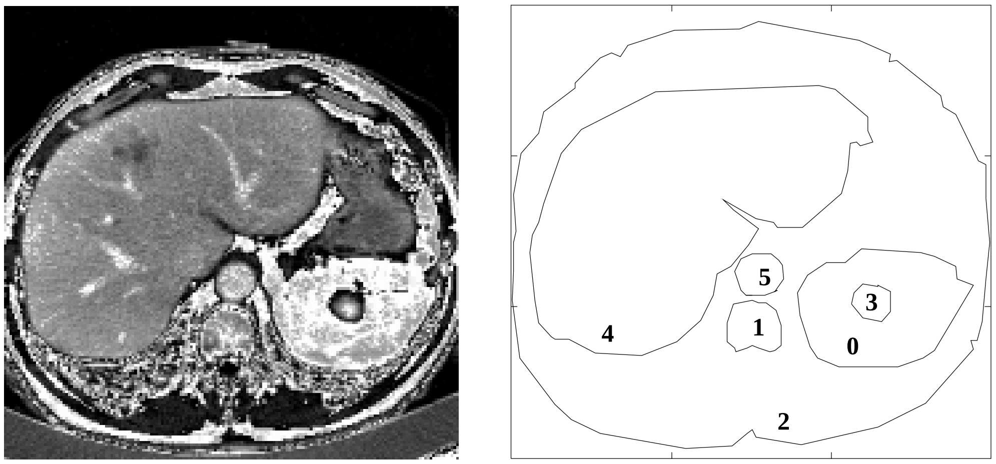


Figure 1: Example of an original grey-level image and its segmented form.

its position, with attribute values. Ordering must be based on criteria that clearly differentiate the objects among them. Position is such a criterion, since objects are usually scattered in the image. However, size or shape do not necessarily provide good ordering criteria, since in certain cases images may contain similar objects. Therefore, we have chosen to base ordering mainly on criteria relating to position.

3.1 Representation of the "Left/Right" and "Below/Above" Relationships

Two ordering criteria are presented and discussed. The first criterion can be used in cases where images are scaled with respect to each other, while the second ordering criterion can be used only in cases where images are at a fixed scale. Ordering is the first step towards producing the string representations of the "left/right" and of the "below/above" relationships between the objects contained in an image subset.

3.1.1 First Ordering Criterion

Let $(a_0, a_1, \ldots, a_{k-1})$, be an image subset of size k, where $k \in [2, \min(n, K_{max})]$. For any two objects a_i, a_j, $i, j \in [0, k-1]$ with centers of mass (x_i, y_i) and (x_j, y_j) respectively, either a_i is a "predecessor" of a_j which is written as $a_i \prec a_j$, or a_i is a "successor" of a_j which is written as $a_i \succ a_j$. Specifically, the first ordering criterion is written as follows:

$$\forall\, i, j \in [0, k-1] \begin{cases} a_i \prec a_j, & \text{if } x_i < x_j \text{ OR, } y_i < y_j \text{ if } x_i = x_j; \\ a_i \succ a_j, & \text{otherwise.} \end{cases} \tag{1}$$

The application of this ordering criterion to the objects contained in an image subset gives rise to a permutation string p, which is actually the ordered sequence of indices corresponding to the above objects. For example, the permutation string p corresponding to the image subset (2 3 4 5) is (4 2 5 3). Henceforth, string p will be used to represent the original image subset itself.

A second permutation string $p^{'}$ is produced by ordering objects according to the following rule:

$$\forall\, i, j \in [0, k-1] \begin{cases} a_i \prec a_j, & \text{if } y_i < y_j \text{ OR, } x_i < x_j \text{ if } y_i = y_j; \\ a_i \succ a_j, & \text{otherwise.} \end{cases} \tag{2}$$

We now define a new entity called "rank". The rank r_i of object p_i is defined as the number of objects preceding it in string $p^{'}$. Given strings p_i and $p^{'}$, r_i is computed as

$$r_i = j \iff p_i = p'_j, \quad 0 \leq i, j < k. \tag{3}$$

Using Equation 3, a rank string r is produced corresponding to the ordered sequence p. For example, the permutation string p' corresponding to the image subset (2 3 4 5) is (3 5 2 4) and its rank string r is (3 2 1 0).

The first ordering criterion guarantees that objects are ordered with respect to the x-axis. The i-th object from the left is object p_i. Moreover, the rank string r is such that r_i denotes the number of objects which are below the i-th object from the left. Therefore the rank string r completely characterizes the relative positions of the objects contained in image subset p. String r defines a permutation over the set $\{0, 1, \ldots k-1\}$, since no two objects have equal ranks. Thus, string r may take $k!$ different values. An ordered image subset p is mapped to a unique address in an address space of size $k!$ by computing the rank (order) of r with respect to a listing of permutations. For example, the rank corresponding to the ordered image subset (4 2 5 3) is 12. The ranking algorithm we used is that by Johnson and Trotter [19].

It has been assumed that no two objects have the same center of mass. The ordering of image subsets containing concentric objects (this may happen when one object contains another) is ambiguous and may result in different representations of subsets with similar spatial relationships. In such cases, the ordering of objects can be based on more image properties in addition to the center of mass. For example, given two objects with the same center of mass, the first ordering criterion can be defined so that the object having the smaller size proceeds the second in the ordered sequence. If the objects have the same size, then a third property (e.g., the roundness), a fourth and so on, can be used so resolve ambiguities related to the order of objects. However, such situations are very rare.

3.1.2 Second Ordering Criterion

Let $(a_0, a_1, \ldots, a_{k-1})$, be an image subset of size k, where $k \in [2, \min(n, K_{max})]$. The image area is partitioned by an $M \times N$ rectangular grid, whose cells are indexed from 0 to $MN - 1$. Each object is assigned a rank or position r_i equal to the index of the grid cell containing its center of mass (x_i, y_i). In particular, the rank r_i of object a_i is computed as follows:

$$r_i = s_{x_i} \cdot N + s_{y_i}, \quad 0 \leq i < k, \tag{4}$$

where $s_{x_i} = \left\lfloor \frac{x_i}{X} \right\rfloor \cdot N$ and $s_{y_i} = \left\lfloor \frac{y_i}{Y} \right\rfloor \cdot M$. X and Y are the actual sizes (in pixels) of the rectangular grid along the horizontal and vertical dimension respectively, while s_{x_i} and s_{y_i} are the x, y coordinates of the position r_i of object a_i with respect to the $N \times M$ rectangular grid ($s_{x_i} \in [0, N-1]$ and $s_{y_i} \in [0, M-1]$). Figure 2 (a) illustrates a 3 × 3 grid of size 256 × 256. In Figure 2 (b), the same grid has been placed over the image of Figure 1. Object 2 has position 4, object 3 has position 7, while both objects 4 and 5 has position 4.

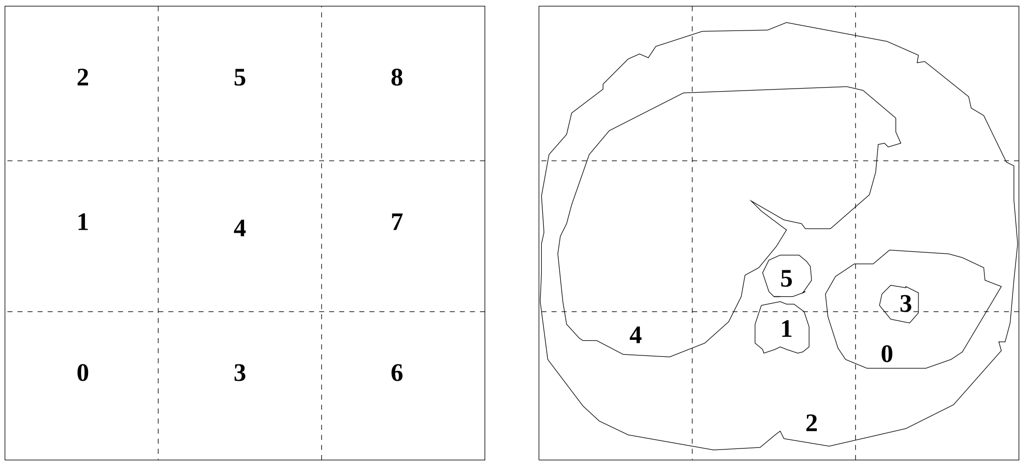


Figure 2: (a) A 3 $\times$ 3 rectangular grid and (b) the same grid placed over the example image of Figure 3.1.

Ordering is based on object positions. However, objects sharing the same grid position cannot be ordered unless a secondary criterion is used. We have chosen to base such a secondary criterion on object properties. In the definition given below, the size of objects is used as a secondary criterion: among two or more objects sharing the same position, ordering is based on size. In particular, for any two objects a_i, a_j, $i, j \in [0, k-1]$, with positions r_i, r_j respectively, the second ordering criterion is written as follows:

$$\forall\, i, j \;\in\; [0, k-1] \begin{cases} a_i \prec a_j, & \text{if } r_i < r_j \text{ OR, } size(a_i) < size(a_j) \text{ if } r_i = r_j; \\ a_i \succ a_j, & \text{otherwise.} \end{cases} \tag{5}$$

If among objects sharing the same grid position there are objects of equal size, then the ordering is ambiguous and a third criterion has to be used (e.g., roundness). In general, the definition of the second ordering criterion can be extended to included any number of criteria (e.g., object roundness, orientation etc.) in addition to the position and the size of objects. However, ambiguities are reduced when grids are dense (e.g., $N, M \geq 3$) and the number of objects contained in image subsets is low (e.g., 2 - 6).

The application of the second ordering criterion to the objects contained in a given image subset gives rise to a permutation string p, which is the ordered sequence of object indices, and a string r of positions (ranks) corresponding to the above ordered sequence of objects. For example, the permutation string p corresponding to the image subset (2 3 4 5) is (5 4 2 3) and the rank string r is (4 4 4 7).

The number of different ways for placing k objects on an $M \times N$ rectangular grid, is equal to the number of “l-part compositions of k”, where $l = NM$. Therefore, the number of different values a string r may take is equal to $\binom{k+l-1}{l-1} = \binom{k+l-1}{k} = \frac{(k+l-1)!}{k! \cdot (l-1)!}$. An ordered image subset p can be indexed in terms of the relative positions between the objects it contains, by computing the rank of string r with respect to a listing of compositions [20]. For example, the index computed to the ordered image subset (5 4 2 3), which has rank string (4 4 4 7), is 244.

3.1.3 Comparison Between the Two Ordering Criteria

Representations derived by the application of the first ordering criterion are both translation and scale invariant (i.e., images translated or scaled with respect to each other result in the same representation). Translation invariance is assured, since only relative positions are taken into account in determining the order of objects. Similarly, scale invariance is assured, since no distance criterion is used. Therefore, when the first ordering criterion is used, the property of distance cannot be used in queries. However, in certain situations, and in particular in cases where all images are at a fixed scale, distance is important in determining the positions of objects. This may be the case when objects which are close enough to each other or the one contains the other must be assigned the same position. In such cases, the second ordering criterion must be used.

Representations derived by the application of the second ordering criterion are neither scale nor translation invariant. Translation invariance can however, be easily achieved by a simple coordinate transformation: first, the minimum enclosing rectangle specified by the centers of mass of the objects contained in p is computed. Let (α_x, α_y) be the coordinates of its lower left corner. The position r_i of each object p_i is then computed according to Equation 4 by using, in place of the pair of coordinates (s_{x_i}, s_{y_i}), the pair $(s_{x_i} - \alpha_x, s_{y_i} - \alpha_y)$.

Both kinds of representations are rather sensitive to image rotations. Rotation invariance can be achieved only in cases where a reference direction can be identified (e.g., specified interactively by the user) so that, prior to any processing, images are rotated and placed to a standard orientation.

3.2 Representation of the Inclusion Relationships

Given an ordered image subset $p = (p_0, p_1, \ldots p_{k-1})$, an inclusion string $w = (w_0, w_1, \ldots w_{k-1})$ representing the "inside/outside" relationships between objects is constructed as follows:

$$\forall\, i \in [0, k-1],\ w_i = \begin{cases} j & \text{if } p_i \text{ is both closer and inside } p_j, \quad 0 \le j < k; \\ i & \text{if } p_i \text{ is contained by the \textit{image frame} only.} \end{cases} \tag{6}$$

An object is contained by another, if all its contour points are contained by the polygonal contour corresponding to the second object [21]. Besides, the minimum distance between all pairs of objects contained in the image under consideration must be computed.

A string w depends on the ordered sequence p, which is turn depends on the ordering criterion applied. A string w may be considered as one of the "k-base representations of k" elements. These are k^k. Therefore, an ordered subset p can be indexed in terms of the inclusion relationships between objects it contains by computing the rank of w with respect to a listing of the k-base representations of k elements. For example, the inclusion string w corresponding to the ordered (according to the first criterion) subset (4 2 5 3) is (1 1 1 1) and has rank 85. The actual size of the address space corresponding to inclusion strings w of size k is less than k^k (i.e., there are addresses which do not correspond to inclusion relationships between k objects). For example, the actual size of the address space corresponding to inclusion strings w of size 2 is 3 ($3^2 = 4$), 16 for strings of size 3 ($3^3 = 27$), 125 for strings of size 4 ($4^4 = 256$), $1,296$ for strings of size 5 ($5^5 = 3{,}125$) and $16{,}807$ for strings of size 6 ($6^6 = 46{,}656$). For further details see [22].

3.3 Representation of Object Properties

The description of individual objects is given in terms of properties corresponding to global object characteristics such as area, perimeter, roundness, orientation with respect to a reference direction, properties of classes etc. Given an ordered image subset p, each of the above properties gives rise to a separate attribute string $(u_0, u_1, \ldots u_{k-1})$, where

$$\forall\, i \;\in\; [0, k-1],\; u_i = \left\lfloor \frac{\text{property value of } p_i}{\text{maximum property value}} \right\rfloor \cdot q. \tag{7}$$

If the property value of p_i equals to the maximum property value, then u_i must be set to $q - 1$.

The "maximum property value" is usually different for different properties. For example, roundness has maximum value 1 (corresponding to a circle), orientation has maximum value $\pi = 3.141\ldots$, etc. In cases where the resulting representation has to be independent of scale (e.g., when the first ordering criterion is used), maximum size values (e.g., perimeter, area) are set equal to the size of the biggest object in the image. Otherwise, maximum size values can be set equal to a reference size (e.g., the size of the image). q is an integer, called "quantization value", corresponding to the number of different values a property may take. In particular, an object is assigned an integer property value in the range $[0, q-1]$. For example, if $q = 3$ the property of orientation takes values 0, 1, 2 corresponding to objects having orientation (with respect to the horizontal direction), between 0 and $\frac{\pi}{3}$, $\frac{\pi}{3}$ and $\frac{2\pi}{3}$, $\frac{2\pi}{3}$ and π respectively.

Properties such as roundness, orientation etc., are computed based on the polygonally approximated bounding contours of objects. In particular, the orientation of an object is defined to be equal to the angle, with respect to the horizontal direction, of the axis of elongation. This is the axis of least second moment. The roundness of an object is defined as the ratio of the smallest to the largest second moment [23]. Moreover, the area of an object may easily be computed from its polygonal contour $((x_0, y_0), (x_1, y_1), \ldots (x_{m-1}, y_{m-1}))$ as $\frac{1}{2}\sum_{i=0}^{m-1}(x_{i+1} \cdot y_i - x_i \cdot y_{i+1})$, where subscript calculations are modulo m (number of contour points). Such computations have time complexity proportional to the number of contour points.

Each attribute string of size k is mapped to a unique address in an address space of size q^k by computing its rank with respect to a listing of the "q-base representations of k" elements. Higher (lower) quantization values increase (decrease) the size of the address space exponentially. However, higher (lower) quantization values increase (decrease) the accuracy of representations. The ordered (using the first ordering criterion) image subset (4 2 5 3) has size (area) string (1 2 0 0) which has rank 7, roundness string (1 2 2 2) which has rank 79 and orientation string (0 0 0 2) which has rank 54. In all cases $q = 3$.

3.4 Stability of Image Representations

Image representations are unique; only subsets having similar properties result in the same attribute strings and mapped to the same indices. However, in certain cases, representations are not tolerant to small variations of property values and become unstable; subsets with slightly different property values may be mapped to different attribute strings and, therefore, different indices.

Instabilities may occur when (a) objects are close enough to each other and the first ordering criterion is used (e.g., objects 0 and 3 of the image of Figure 1) or (b) they have centers of mass near the borders of a grid cell (e.g., object 3) and the second ordering criterion is used. A slight translation of such an object along the x or the y direction would change the ordered sequence of object indices. To deal with such situations, one must consider all possible orders of sequences containing objects which are very close to each other. For example, for the image subset consisting of objects 0, 3 and 4, two ordered subsets, (4 3 0) and (4 0 3), must represented and stored separately. In addition, object 3 of the right image of Figure 2 must be considered as having both positions 6 and 7. An image subset containing object 3, e.g., (5 4 3), must then be represented and stored twice, once with rank string (4 4 6) and once with rank string (4 4 7).

Instabilities may also occur when objects have continuous property values approximately equal to the threshold values. For example, for the property of roundness and a quantization value $q = 3$, the threshold values are $0.33\ldots$ and $0.66\ldots$. Object 4 in Figure 2 has roundness $0,33$ and may take a discrete value of 0 or 1. The representation of image subsets containing object 4 is unstable. Such objects must be assigned two discrete property values and an image subset containing such an object must be represented and stored twice. For example, the image subset (4 1 5) must be represented and stored twice, once with roundness string (0 2 2) and once with roundness string (1 2 2).

3.5 Completeness of Image Representations

The proposed representations succeed in capturing image content in cases of images consisting of disjoint objects having simple shapes. However, such representations cannot be used when images contain occluded objects. Global object characteristics (e.g., size, roundness) change drastically and become unreliable if substantial parts of objects are hidden; Similarly, such representations cannot capture information related to details of object shapes. However, in each of the above two cases, such representations can be used to index the images of an IDB. In particular, even if objects are partially visible, if object positions are identified correctly (e.q., specified by the user), indexing can still be based on relationships. In addition, even if objects have complex shapes, indexing can still be based on global object properties. Retrievals respond with a set of candidate images which can then be compared against the query on the basis of additional properties, either relational or specific to the shape of objects they contain, using techniques such as those proposed in [24, 25, 26].

Object positions play an important role in deriving reliable image representations and must be specified precisely. So far, it has been assumed that object positions are defined by their center of mass. In cases where the true center of mass cannot be obtained of falls outside the object contour, a point which is representative of the object position can be specified by an expert user.

4 Image Indexing and Storage

The computed representations of subsets derived from all images are logical images and are stored in a logical database. The logical database consists of a set $(H_2 \ldots, H_{K_{max}})$ of files, where K_{max} is the maximum size of image subsets under consideration. The image subsets of size $k,\ 2 \leq k \leq K_{max}$, together with their representations are all stored in file H_k. Each image subset p (which is the ordered sequence of object indices) is represented by a tuple of strings of the form $(r, w, \ldots)$ where, r is its corresponding rank string representing the "left/right" and "below/above" relationships between objects, w is the inclusion string, and the remaining strings correspond to properties of individual objects. Three such strings are used: s to encode the size property, c to encode the roundness property and o to encode the orientation property of those objects whose indices are in p. Therefore, the representation of an image subset p is given by a tuple of the form (r, w, s, c, o). In order to satisfy the needs for efficient storage and access of image subsets, an addressing scheme is proposed and a file structure specific to the above addressing scheme are introduced.

4.1 Proposed Addressing Scheme

An image subset is considered to be the basic entity in the proposed indexing scheme. In particular, images are indexed based on representations of the set of all derived subsets. An image containing n objects is decomposed into the set of ordered image subsets $\left\{p_k^l \mid 2 \leq k \leq \min(n, K_{max}),\ 0 \leq l < \binom{n}{k}\right\}$. The representation of an image subset p_k^l takes the form $(d_k^{l,0}, d_k^{l,1}, \ldots, d_k^{l,\nu-1})$, where ν is the number of attribute strings and $d_k^{l,i}, 0 \leq i < \nu$, is the address computed to the i-th attribute string. In particular, the (r, w, s, c, o) representation of the l-th subset of size k can be written as $(d_k^{l,0}, d_k^{l,1}, d_k^{l,2}, d_k^{l,3}, d_k^{l,4})$ or $(d_k^{l,r}, d_k^{l,w}, d_k^{l,s}, d_k^{l,c}, d_k^{l,o})$ (for clarity, the index i has been substituted by the symbol of its corresponding attribute string). p_k^l can then be mapped to a single address I_k^l: $(d_k^{l,0}, d_k^{l,1}, \ldots, d_k^{l,\nu-1})$ is considered to be the representation of I_k^l in the "mixed radix system" $(D_k^0, D_k^1, \ldots, D_k^{\nu-1})$, where $D_k^i,\ 0 \leq i < \nu$, is the size of the address space corresponding to the i-th attribute string. The index I_k^l is computed as follows:

$$I_k^l = d_k^{l,0} + d_k^{l,1} \cdot D_k^0 + d_k^{l,2} \cdot D_k^0 \cdot D_k^1 + \cdots + d_k^{l,\nu-1} \cdot \prod_{i=0}^{\nu-2} D_k^i, \tag{8}$$

where $2 \leq k \leq \min(n, K_{max}), \quad 0 \leq l < \binom{n}{k}$. Each $d_k^{l,i}$ satisfies $0 \leq d_k^{l,i} < D_k^i$ and $d_k^{l,\nu-1} \neq 0$, except for $I_k^l = 0$. The size of the address space specified by Equation 8 is equal to

$$D_k = \prod_{i=0}^{i=\nu-1} D_k^i, \quad 2 \leq k \leq K_{max}. \tag{9}$$

The size of the address space is equivalent to disk space (see section 4.3). D_k becomes extremely large for relatively large values of k and ν (e.g., for $k > 4$ and $\nu > 2$, D_k takes values in the order of 10^9). One way to deal with large address spaces is to allow "collisions", in which cases, two or more image subsets with different representations are mapped to the same address. If $DIRSIZE$ is the maximum size of the address space allowed (e.g., $DIRSIZE = 10^6$), a mapping that allows collisions is

$$A_k^l = I_k^l \bmod DIRSIZE, \quad B_k^l = \left\lfloor \frac{I_k^l}{DIRSIZE} \right\rfloor, \tag{10}$$

where $2 \leq k \leq \min(n, K_{max}), \quad 0 \leq l < \binom{n}{k}$. An image subset p_k^l is now characterized by the pair (A_k^l, B_k^l). A_k^l is a new address, called "primary index" and takes values in the range $[0, DIRSIZE - 1]$. B_k^l is a second address, called "secondary index", which takes values in the range $\left[0, \left\lfloor \frac{D_k}{DIRSIZE} \right\rfloor\right]$ and is used to distinguish among image subsets with the same primary indices.

The size of the address space decreases when $\kappa < \nu$ attribute indices are used in Equation 8 to compute I_k^l. Such indices correspond to attribute strings and image properties which are called “primary attributes” and “primary properties” respectively. The $\nu - \kappa$ remaining attribute strings, which are not primary, and their corresponding image properties are called “secondary” and are used to compute a secondary index C_k^l according to Equation 8. Therefore, an image subset p_k^l is represented by a triple (A_k^l, B_k^l, C_k^l) consisting of its corresponding primary and secondary indices.

4.2 Selection of Primary Attributes

The primary attributes must correspond to the most discriminant image properties (see Section 5.3). In the optimum case, primary indices have a uniform distribution and each index holds the same number of image subsets. The selection of primary attributes is based on measurements obtained from a prototype set of images which are considered to be characteristic of the application under consideration. In particular, for the i-th attribute string of a given image representation consisting of ν attributes and for the indices corresponding to all attribute strings of size $k \in [2, K_{max}]$, we compute the variance σ_k^i as

$$\sigma_k^i = \frac{\sum_{j=0}^{D_k^i - 1} \left(N_{k,j}^i - \overline{N_k^i} \right)^2}{D_k^i - 1}, \quad 0 \leq i < \nu, \quad 2 \leq k \leq K_{max}, \tag{11}$$

where D_k^i is the address space corresponding to image subsets of size k and the i-th attribute string. $N_{k,j}^i$ is the number of attribute strings having index j. $\overline{N_k^i}$ is their mean value computed over D_k^i as $\overline{N_k^i} = \frac{\sum_{j=0}^{D_k^i - 1} N_{k,j}^i}{D_k^i}$. The variance measures the actual amount of variation of a set of data and depends on the scale of measurement: the variance σ_k^i is computed with respect to the address space D_k^i which varies for different attribute strings of the same size k. Therefore, the variance is computed with respect to mean values which vary depending on the kind of the attribute under consideration. To compare the variation of several sets of data we use the “coefficient of variation” CV_k^i, which gives the variation as a percentage of the mean values $\overline{N_k^i}$ which in turn are normalized with respect to D_k^i. In particular, CV_k^i is computed as

$$CV_k^i = \frac{\sqrt{\sigma_k^i}}{\overline{N_k^i}} \cdot 100, \quad 0 \leq i < \nu, \quad 2 \leq k \leq K_{max}. \tag{12}$$

The coefficient of variation CV_k^i is computed for all attributes and for all string sizes $k \in [2, K_{max}]$. For a specific size k, the most discriminant attributes are those having the smallest coefficient of variation. Therefore, different primary properties may correspond to subsets of different size k. Based on the computed values of coefficient of variation, attribute strings and corresponding image properties can be ordered in terms of decreasing discriminative power. A number of attributes (e.g., 2 or 3), of the first in this ordered sequence, are then chosen to be the primary attributes. The specification of the number of primary attributes is another important problem which is discussed separately in Section 5.4.

In developing a prototype IDB which supports the indexing of images by content we used 226 computed tomographic (CT) and magnetic resonance (MR) images. We considered this set of images to be characteristic of the application domain under consideration and we used them

Table 1: Coefficient of variation computed for all attribute indices corresponding to image subsets of size $k \in [2, 6]$, derived from a set of medical images.

k	CV_k^r	CV_k^w	CV_k^c	CV_k^s	CV_k^o
2	9.01	109.37	15.20	161.73	39.30
3	10.90	251.72	35.62	274.55	49.32
4	22.42	765.55	59.42	452.92	69.38
5	48.02	2,601.81	98.89	747.55	108.32
6	121.60	10,402.70	280.21	1,254.8	195.49

for the selection of primary attributes. Table 1 shows the values of the coefficient of variation computed for five attribute strings, namely the r, w, s, c and o, having size $k \in [2, 6]$ (for clarity, the index i in CV_k^i has been substituted by the symbol of its corresponding attribute string). We used the first ordering criterion and a quantization value of $q = 3$. In the computation of CV_k^w, the actual size of the address space D_k^w has been taken into account (see Section 3.2). According to Table 1, r is the most discriminating attribute followed by c, o and w. In particular, the ordered sequence of attributes in order of decreasing discriminating power is (r, c, o, s, w).

Henceforth, the roundness property of objects represented by string c, together with the "left/right" and the "below/above" relationships of objects represented by string r, are considered to be primary image properties and primary attributes respectively and are used to compute the primary indices. The remaining three attribute strings, namely the o, s and w corresponding to the properties of orientation, (relative) size and the inclusion relationships respectively are considered to be secondary and are used to compute secondary indices. For example, the ordered image subset (4 2 5 3) derived from the image of Figure 1 is represented by the triple $(1{,}908,\ 0,\ 562{,}066)$ of primary and secondary indices.

4.3 Proposed File Structure

All image subsets of size k, $k \in [2, K_{max}]$, are stored in the H_k file. A file H_k consists of a set of "data pages" of fixed capacity. In particular, for each subset p_k^l having index A_k^l, a tuple $(image, p_k^l, B_k^l, C_k^l)$ is stored, where "image" is the name of the image from which p_k^l has been derived and acts as a pointer to the original image file stored in the physical database. Each page stores image subsets with the same primary index. A pointer is kept representing the association between page addresses on disk and primary indices. In fact, an array of such pointers is maintained, called "directory". The size of a directory is computed according to Equation 9. A directory entry may also contain a second address field pointing to the next free position within a page.

An attempt to insert an image subset in an already full page causes an "overflow". Overflows are handled by chaining: a new page is allocated (in which the image subset is stored) and linked with the overflowed page. New pages are always inserted at the beginning of the list so that there is no need to search the whole list of pages in order to locate a nonempty page. Before any data are stored, an "empty" directory (i.e., a directory having all its pointers set to null) is maintained for each H_k file. Figure 3 illustrates a file structure with the above characteristics.

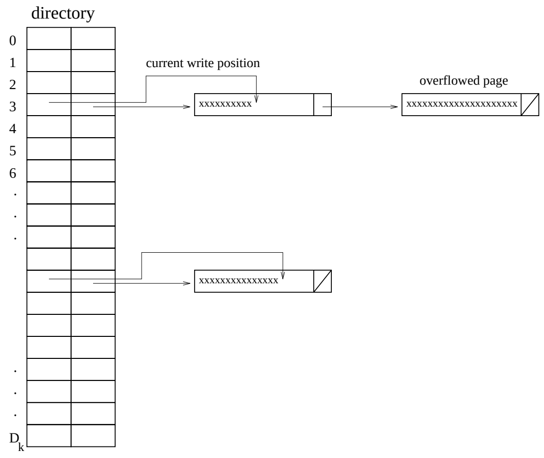


Figure 3: File structure H_k with directory of size D_k. Unused directory pointers are null.

The advantage of this organization is that it provides very fast direct access on the basis of a given address. It has the disadvantage that updates (i.e., modifications of stored data) and deletions are difficult to be performed. However, insertions and retrievals are the two more common operations that take place in an IDB; updates and deletions are rather rare.

4.4 Storage Requirements

The storage space required by each H_k file structure is

$$S_{N,k} = b_1 \cdot \sum_{i=1}^{N} \binom{n_i}{k} + b_2 \cdot \min(D_k, DIRISIZE), \quad 2 \le k \le \min(n_i, K_{max}). \tag{13}$$

Therefore, the storage space for the IDB as a whole is $S_N = \sum_{k=2}^{K_{max}} S_{N,k}$. The first term in Equation 13 relates to the space storing the actual image data (image subsets). It is dynamic, since it increases with the number of images stored in the IDB. N is the number of images stored, n_i is the number of objects contained in the i-th image and b_1 is the size of information stored per image subset ($b_1 = 15$ bytes on the average). The second term of Equation 13 relates to the data space required to store the directories. It is static, since it never changes no matter how large or small is the number of images stored. b_2 is the amount of space required to implement the two directory pointers (usually $b_2 = 8$ bytes).

The number of image subsets produced, and thus the amount of data stored, can be very large especially when the number of objects contained in images is large (e.g., more than 10). It can become even larger when images are noisy and segmentations are inexact. To avoid such difficulties and to keep the amount of data manageable, segmentations need to be carried out interactively so that insignificant segments and segments resulting from noise are deleted. In this case and for the kinds of images found in most applications, the average number of objects stored per image can be kept small (typically less than 10). The number of image subsets stored per image, and thus the amount of space required, increases with K_{max}. Figure 4 shows the number of image subsets produced from an image as a function of the number of objects it contains, for (a) $K_{max} = 2$, (b) $K_{max} = 3$, (c) $K_{max} = 4$, (d) $K_{max} = 5$ and (e) $K_{max} = 6$. If we consider $b_1 = 15$ bytes and $K_{max} = 6$, then images containing up to 10 objects require

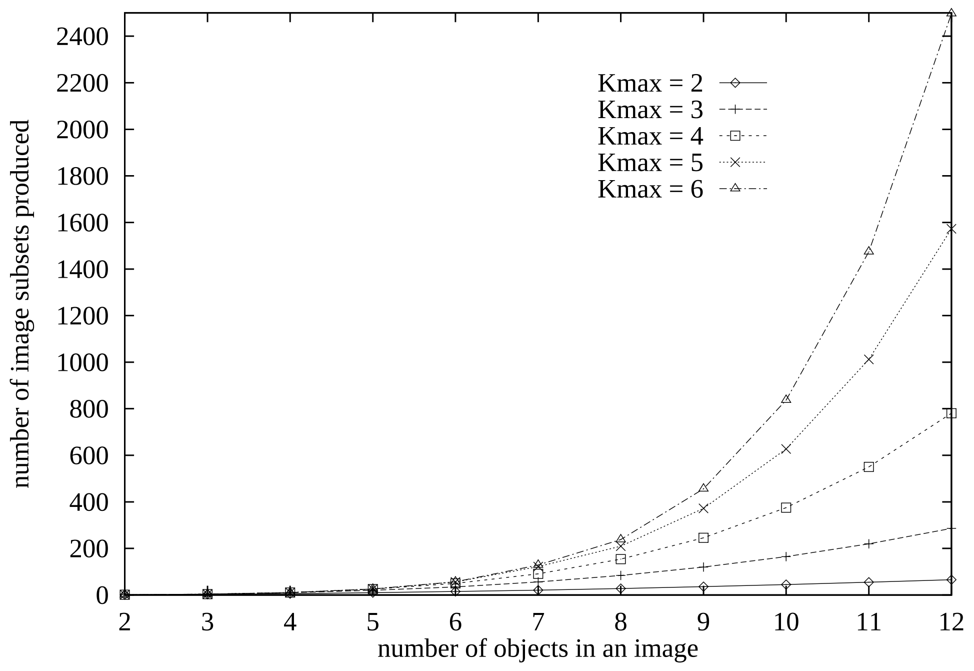


Figure 4: Number of image subsets produced from an image, as a function of the number of objects it contains, for (a) $K_{max} = 2$, (b) $K_{max} = 3$, (c) $K_{max} = 4$, (d) $K_{max} = 5$ and (e) $K_{max} = 6$.

less than $800 \cdot 15 = 12{,}000$ bytes for storage. Images containing more than 10 objects require much more space for storage.

For the prototype IDB consisting of 226 medical images and for $K_{max} = 6$, there are $16{,}404$ stored image subsets. Among them, there are $2{,}557$ subsets of size 2, $3{,}917$ subsets of size 3, $4{,}286$ subsets of size 4, $3{,}480$ subsets of size 5 and $2{,}163$ subsets of size 6. Assuming $b_1 = 15$ bytes on the average, the average required amount of storage space per stored image subset is $1{,}088$ bytes (the amount of storage space corresponding to the directories has not been taken into account).

5 Image Retrieval

All queries address the logical database rather than the raw image data stored in the physical database. We concentrate our attention on the case of queries by image example: a query image or a sketch of image segments is given, it is analyzed and a representation similar to those of the images stored in the IDB is created. Sketches corresponding to all images matching the query are retrieved and displayed. The user can then specify which (original) images are going to be retrieved from the physical database using an image browser. Representations of image subsets of equal size having the same property strings with those of the query are also retrieved and displayed.

5.1 Image Similarity Criteria

Let p_m be an image consisting of m objects and let $(d_m^0, d_m^1, \ldots, d_m^{\nu-1})$ be its representation corresponding to a set of ν properties. Furthermore, let p_k be an image subset obtained from an image consisting of n objects and let $(d_k^0, d_k^1, \ldots, d_k^{\nu-1})$ be its representation. The image p_m is "similar" to the image subset p_k or p_m "matches" p_k, in which case we write $p_m \sim p_k$, if the following condition holds:

$$p_m \sim p_k \iff k = m \wedge d_m^i = d_k^i, \quad \forall\, i \in [0, \nu - 1]. \tag{14}$$

Equivalently, an image is similar to an image subset if they have both equal size and same property strings, or they have the same representation of primary and secondary indices. If p_m matches p_k, then the i-th ($i \in [0, m-1]$) object of p_m matches the i-th object of p_k. Furthermore, an image p_m matches an other image consisting of n objects if, $m \leq n$ and p_m matches at least one of the image subsets of size m generated from the image under consideration. If p_m is a query image, its corresponding answer set contains all image subsets which are similar to it.

5.2 Search Strategy

Query processing depends on m, the number of objects contained in the query image. In particular, we distinguish between "direct access" queries corresponding to $2 \leq m \leq K_{max}$ and "indirect access" queries corresponding to $m > K_{max}$, where K_{max} is the maximum size of image subsets stored. The search is performed in two steps, namely "hypothesis generation" and "hypothesis verification". During the first step, a number of candidate image subsets (or images) matching the query are retrieved. During the second step, the final answer set containing image subsets (or images) matching the query is constructed.

5.2.1 Direct Access Queries: $2 \leq m \leq K_{max}$

- **Hypothesis generation:** the primary and secondary indices of the query image are computed first (see Section 4.1). The query addresses the H_m file and all image subsets with the same primary index are retrieved.
- **Hypothesis verification:** all hypothesized image subsets are matched (one by one) against the query with respect to the secondary indices.

5.2.2 Indirect Access Queries: $m > K_{max}$

- **Hypothesis generation:** given a query image specifying m objects, all subsets consisting of K_{max} objects are generated, thus creating $\rho = \binom{m}{K_{max}}$ new queries. Each of these queries performs as a direct access query on the $H_{K_{max}}$ file and produces an answer set consisting of names or indices corresponding to images which are similar to it. Therefore, ρ answer sets namely $S_0, S_1, \ldots S_{\rho-1}$, are produced. Their intersection $S = S_0 \cap S_1 \ldots \cap S_{\rho-1}$ is then obtained and used to hypothesize the existence of images matching the original query.

- **Hypothesis verification:** the images contained in the set S are retrieved[2], all image subsets corresponding to retrieved images are generated, their representations are computed, and they are matched against a similar computed representation of the original query image.

5.3 Performance Evaluation

Response time is one possible measure of the performance of retrievals. However, response time depends on characteristics of the particular implementation and of the hardware used. Response time depends on the size of the hypothesized answer set and increases with it. The size of the hypothesized answer set is independent of characteristics of the implementation and can be used as a second measure of the performance of retrievals in addition to response time. It is computed as the percentage of images (or image subsets) retrieved with respect to the total number of images (or image subsets of the same size) stored.

The size of the hypothesized answer set returned in response to queries specifying k objects, is always inverse proportional to the size of the address space D_k. In particular, as D_k increases, the stored image subsets are distributed over a larger space and the number of image subsets stored per index decreases. The evaluation of the performance of retrievals must be based on a mathematical formal model of the distribution of the stored image subsets, which in addition to D_K, takes into account the content of image subsets stored (which in turn depends on the content of images of the application domain under consideration), query content and the content interdependences between image subsets derived from the same image (two or more image subsets may differ by one object). However, such a mathematical formal model cannot be easily derived.

In the special case where all indices appear with the same probability, the size of the hypothesized answer set is equal to $\frac{1}{D_k}$. We attempt to achieve a uniform distribution by selecting as primary, those attributes which distribute the stored image subsets uniformly over their corresponding address space (see Section 4.2). If the distribution is uniform, we can ignore the dependences of performance on the application domain, query content (the performance is fairly the same regardless of query content) and the interdependences between image subsets derived from the same image. In studying the performance of retrievals, independence of query content is achieved by taking the average performance of a large number of queries (i.e., more than 10). Henceforth, in order to keep the analysis of the performance tractable, we consider performance as being only a function of the size of the address space D_k. Specifically:

1. D_k depends on the size k of image subsets stored and increases with it. Therefore, the performance of retrievals improves with the number of query objects, since queries address files storing image subsets of equal size. In particular, the performance of indirect access queries depends on the performance of direct access queries specifying K_{max} objects. Therefore, the performance of indirect access queries improves with K_{max}.

2. D_k depends on the number of primary attributes used and increases with it. Therefore, the performance of retrievals improves with the number of primary attributes. If the number of primary attributes is equal to the total number of attributes of an image representation,

[2]Instead of original images, their polygonal segment forms are retrieved.

then the size of the hypothesized answer set becomes equal to the size of the final answer set.

3. D_k depends on the quantization value and increases with it. Greater quantization values not only increase the accuracy of image representations, but also improve the performance of retrievals.

4. D_k depends on the ordering criterion which has been applied. In particular, the size of the address space corresponding to the first ordering criterion depends only on the size of image subsets. The size of the address space corresponding to the second ordering criterion also depends on the size of the rectangular grid which has been used and increases with it. In general, the second ordering criterion results in larger address spaces and achieves better performance.

5.3.1 Experimental Results

Evaluations have been carried out using a prototype IDB consisting of $1{,}000$ simulated images each containing 2 to 10 objects. For each $n \in [2, 10]$, there are over than 100 images containing n objects. The number of stored image subsets depends on K_{max}. For $K_{max} = 6$, there are $18{,}459$ stored subsets of size 2, $36{,}856$ stored subsets of size 3, $51{,}439$ subsets of size 4, $51{,}274$ subsets of size 5 and $36{,}528$ subsets of size 6. All images are of size 256×256 and they are not scaled with respect to each other. The second ordering criterion has been applied using a rectangular grid of size 3×3. The quantization value q is set to 3 for all object properties. The IDB has been implemented on a magnetic disc connected to a host computer (SUN 4/280). Therefore, the time delays due to data transfers through a communications network are zero.

Queries are distinguished based on the number m of objects they specify. Measurements of both the size of the answer sets and of the retrieval response times have been obtained. To obtain average performance measures, for each value of m ranging from 2 to 6, 20 image queries have been applied and the average performance to queries specifying an equal number of objects has been computed. By applying the methodology of Section 4.2 we found that the ordered sequence of attributes in order of decreasing discriminating power is (r, c, o, s, w).

Query response times account for the time spent in computing the query representation plus the time spent in searching the database and retrieving image data. The later, characterizes the performance of the search mechanism which has been applied and, henceforth, will be used in performance evaluations.

Performance of Direct Access Queries. First we study direct access queries and we set $K_{max} = 6$. Figure 5 shows the average size of the answer sets obtained, as a percentage of image subsets retrieved, plotted against the number of query objects. The relative positions r and the roundness c of objects have been used as primary attributes. For queries specifying 2 objects, the size of the hypothesized answer set (i.e., percentage of image subsets matching the query with respect to r and c) is 5% on the average. Queries become more specific and the size of an answer set decreases as the number of query objects increases. In particular, the size of an answer set drops to 0% for queries specifying more than 5 objects. The same hold in the case of Figure 6 which shows the average size of the answer sets obtained, as a percentage of images retrieved, plotted against the number of query objects.

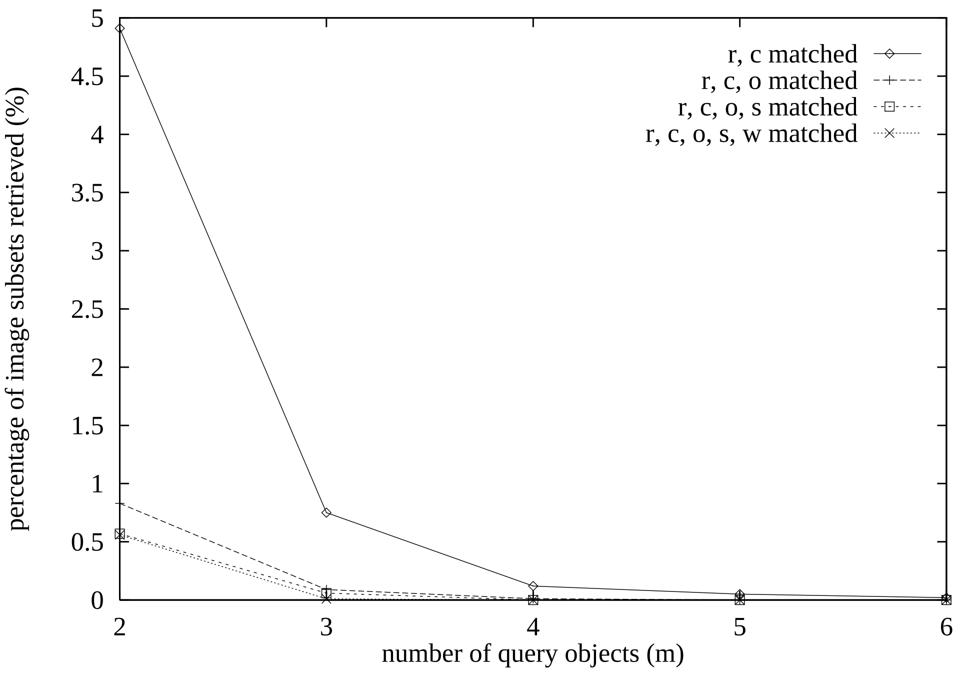


Figure 5: Average size of answer set corresponding to direct access queries, as a percentage of image subsets retrieved, plotted against the number of query objects.

Similarly, queries become more specific and the size of an answer set decreases when comparisons between hypothesized image subsets (or images) and queries are based on more attributes. In comparisons, attributes are used in order of decreasing discriminating power. As shown in Figure 5 and Figure 6, comparisons have been performed based, first on the position r and the roundness c properties of objects (in which case the hypothesized answer sets are obtained), then on the orientation o, then on the size s and, finally, on the inclusion w properties of objects (in which case the final answer sets are obtained).

The response time accounts for the time spent in retrieving the hypothesized answer set plus the time spend in processing the retrieved image data. All retrieved image subsets are compared against the query on the basis of their corresponding secondary indices. The processing takes place in the main memory and is very fast. Therefore, response time accounts mainly to the time spend for retrievals. Figure 7 shows the retrieval response times as a function of the number of query objects. Response times decrease with the number of query objects, since the size of the answer sets obtained and thus the amounts of data to be processed, decrease too. Similarly, query responses are faster when more attributes are used as primary. The response times corresponding to (a) primary attributes r and c, and (b) primary attributes r, c and o are demonstrated. If instead of two, three primary attributes are used, the response time is not significantly reduced and becomes minimum for queries specifying more than 3 objects.

As the number of primary attributes increases, directory sizes increase also and stored data (i.e., image subsets) are distributed over larger address spaces. Therefore, the amounts of data which are retrieved and processed decrease. For example, for primary attributes r and c and for queries specifying 4 objects, the size of the hypothesized answer set is approximately 0.1% on

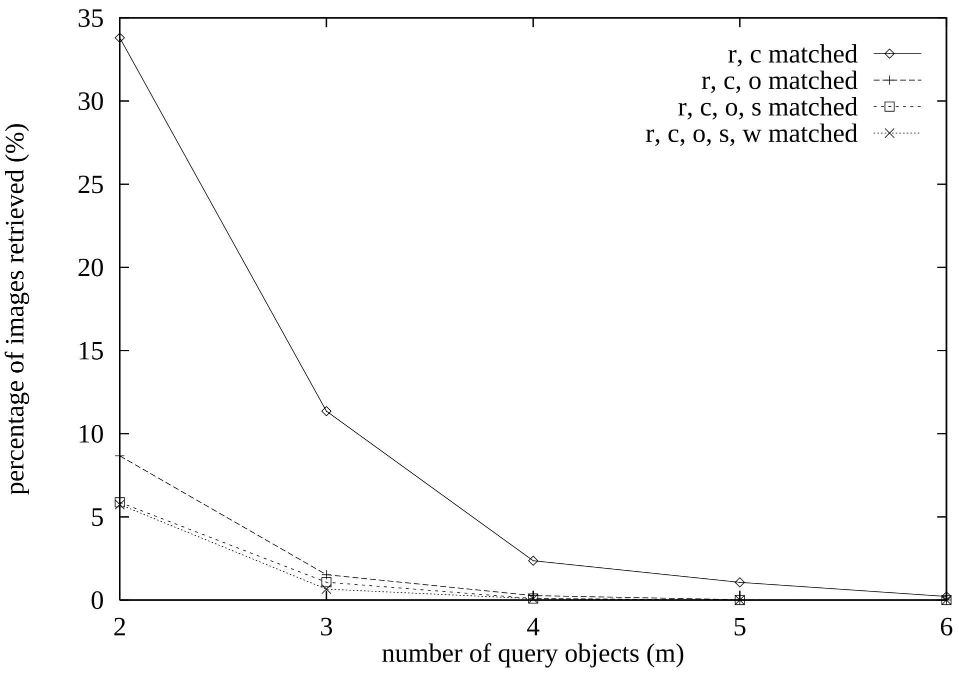


Figure 6: Average size of answer set corresponding to direct access queries, as a percentage of images retrieved, plotted against the number of query objects.

the average. Only $0.1 \cdot 51{,}439/100 \approx 52$ image subsets are retrieved. If the amount of space per stored subset is $b_1 = 15$ bytes, then approximately 770 bytes are retrieved. Such amounts of data can be read in one disc access. Increasing the size of the directory, will result decreasing the amount of retrieved data. However, one disc access is still necessary and the speedup is minimum.

Performance of Indirect Access Queries. An image query specifying $m > K_{max}$ objects, is decomposed into $\rho = \binom{m}{K_{max}}$ query subsets. Each query subset performs as a direct access query on the $H_{K_{max}}$ file and produces an answer set S_i, $0 \leq i < \rho$, consisting of names or indices corresponding to images which are similar to it. The size of each S_i is, on the average, equal to the average size of the final answer set obtained to direct access queries specifying K_{max} objects. The hypothesized answer set S is computed as $S = S_0 \cap S_1 \ldots \cap S_{\rho-1}$. The size of S is, on the average, less than the size of each S_i but, greater than the average size of the final answer set corresponding to direct access queries specifying m objects (i.e., S may contain images which do not match the original query). Let $F_{images}(m)$, $m \in [2, K_{max}]$, be the size of the answer set of direct access queries specifying m objects. S is such that $F_{images}(m) \leq sizeof(S) \leq F_{images}(K_{max})$.

The size of S for $K_{max} = 2$ and $K_{max} = 3$ is shown in Figure 8. The size of F_m for $m \in [2, 6]$ ($K_{max} = 6$), is also shown and corresponds to the curve of Figure 6 with all attributes matched. The size of S decreases with K_{max}: the evidence that an image in S matches the original query is stronger for higher values of K_{max}. Furthermore, the size of S approaches to $F_{images}(m)$ as the number of query objects and the number of properties used in image comparisons increases.

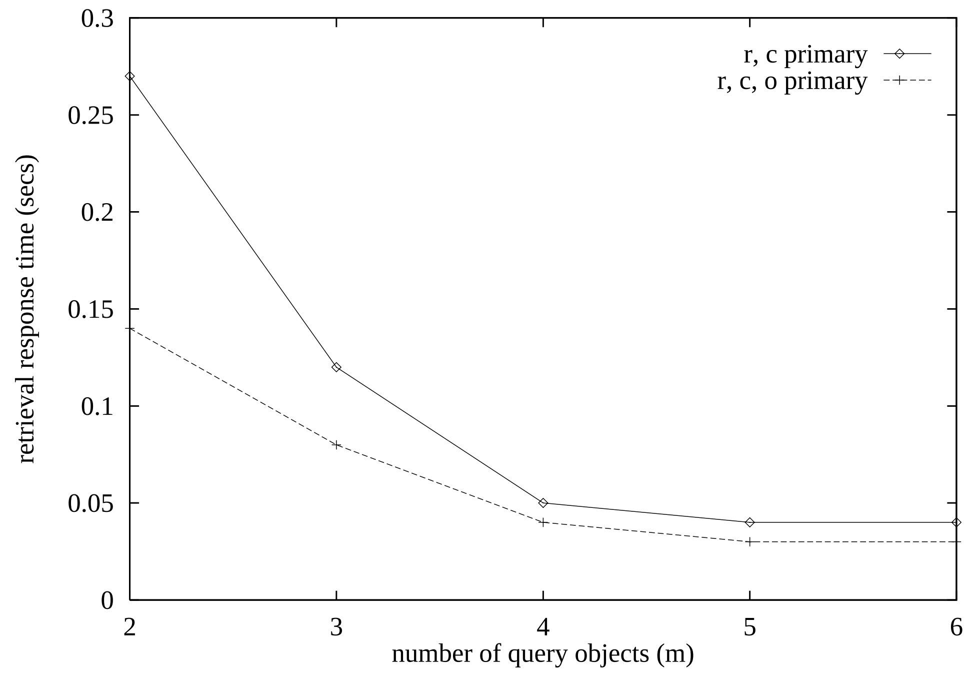


Figure 7: Average retrieval response times corresponding to direct access queries, plotted against the number of query objects.

For example, the size of S becomes approximately equal to F_{images} for $K_{max} \geq 4$ and $m > 4$. So far, we have been using 5 properties, namely the r, c, o, s and w, which have been proven effective in quickly narrowing down the search space in the case of indirect access queries.

All images contained in S are then retrieved and matched against the original query. By comparison with direct access queries, indirect access queries incur an overhead which is due to: (a) the retrieval and processing of intermediate query results and (b) the retrieval of the images contained in S, the processing and the matching of these images against the original query. Therefore, indirect access queries respond slower than direct access queries specifying the same number of objects. The retrieval response times corresponding to indirect access queries for (a) $K_{max} = 2$, (b) $K_{max} = 3$ and (c) $K_{max} = 4$ are shown in Figure 9. The response times corresponding to direct access queries ($K_{max} = 6$) are also shown and correspond to the curve of Figure 7 with r and c as primary attributes. The response times for $K_{max} \geq 4$ and $m > 4$ approach the response times of direct access queries. For such queries, S contains only a few images and the overhead is minimum. When $K_{max} = 2$, as proposed in [12], the overhead is maximum.

5.4 Tailoring Parameters to an Application

Two parameters must be specified before an IDB is put to work: the number of primary attributes and the maximum size of stored image subsets K_{max}. It has been shown that the performance of retrievals improves both with the number of primary attributes and with K_{max}. However, the amount of storage space increases with both of these parameters. The specification

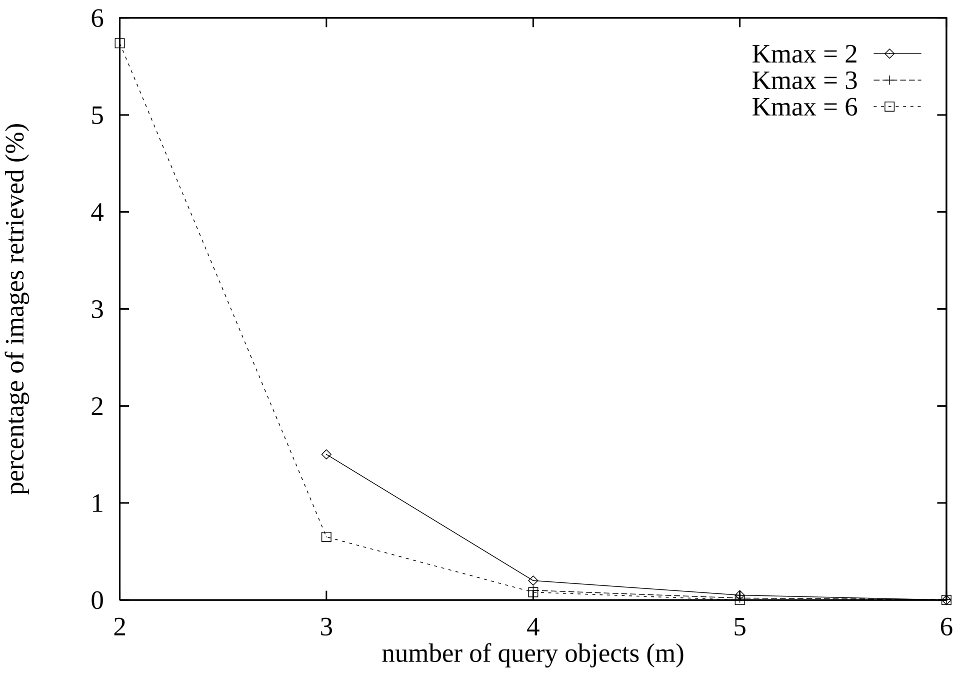


Figure 8: Average size of hypothesized answer set corresponding to indirect access queries, as a percentage of images retrieved, plotted against the number of query objects.

of these parameters depends on the application domain, the content of images involved, and the number of images stored. The IDB of simulated images has been used for the specification of both the number of primary attributes and K_{max}.

5.4.1 Specification of the Number of Primary Attributes

According to Figure 7, increasing the size of directories by using more primary attributes, speeds-up retrievals in cases of queries specifying 2 or 3 objects. In this case, we may choose 3 primary attributes. In cases of queries specifying more than 3 objects, the speedup is minimum. Such queries address directories which are large enough even when 2 primary attributes are used. In this case, we may choose 2 primary attributes. For larger IDBs, the selection of the number of primary attributes can be performed by scaling the retrieval response times accordingly.

5.4.2 Specification of K_{max}

K_{max} must be such that the performance of indirect access queries approaches the performance of corresponding direct access queries specifying the same number of objects. Based on Figure 9 we must choose $K_{max} = 4$ or $K_{max} = 5$. The performance of indirect access queries in terms of the retrieval response times, and therefore the specification of K_{max} based on it, depends mainly on the implementation (i.e., how fast an image is retrieved, how fast its representation is computed, how the intermediate query results are processed etc.). So far, the processing of indirect access queries, the retrieval and the computation processes has not

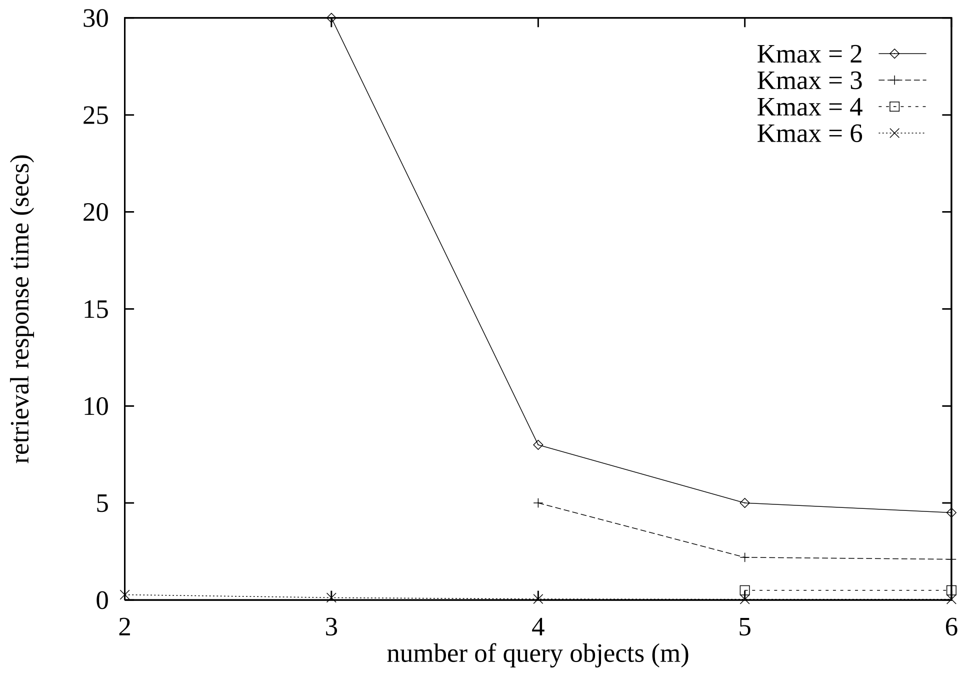


Figure 9: Average retrieval response times corresponding to indirect access queries, plotted against the number of query objects.

been optimized. Based on Figure 8 we may choose $K_{max} = 3$. For $K_{max} = 3$ the size of the hypothesized answer set S becomes approximately equal to the size of the final answer set corresponding to direct access queries specifying the same number of objects. Therefore, there is space to reduce K_{max} from 5 down to 3 by optimizing the processing of indirect access queries.

The performance of indirect access queries depends also on the number of objects contained in images and on the number of properties involved in an image representation. In particular, for images consisting of more than 10 objects on the average, the number of image subsets which are derived and stored for a given value of K_{max} will be larger (see Section 4.4). In this case, the size of the hypothesized answer set S will be larger too. The same will happen when images are compared on the basis of a smaller number of properties (e.g., 1 to 3). In such cases, K_{max} may take greater values.

6 Retrievals on a Medical IDB

The proposed methodology has also been used for the retrieval of images in an IDB consisting of 226 computed tomographic (CT) and magnetic resonance (MR) images. These are images of various parts of the body (e.g., head, abdomen etc.), each consisting of one or more objects surrounded by an outer contour (e.g., skull). All images contain 2 to 8 objects. The images have been obtained at various scales. To obtain scale independent representations the first ordering criterion has been applied. The size values of objects are normalized with respect to the size

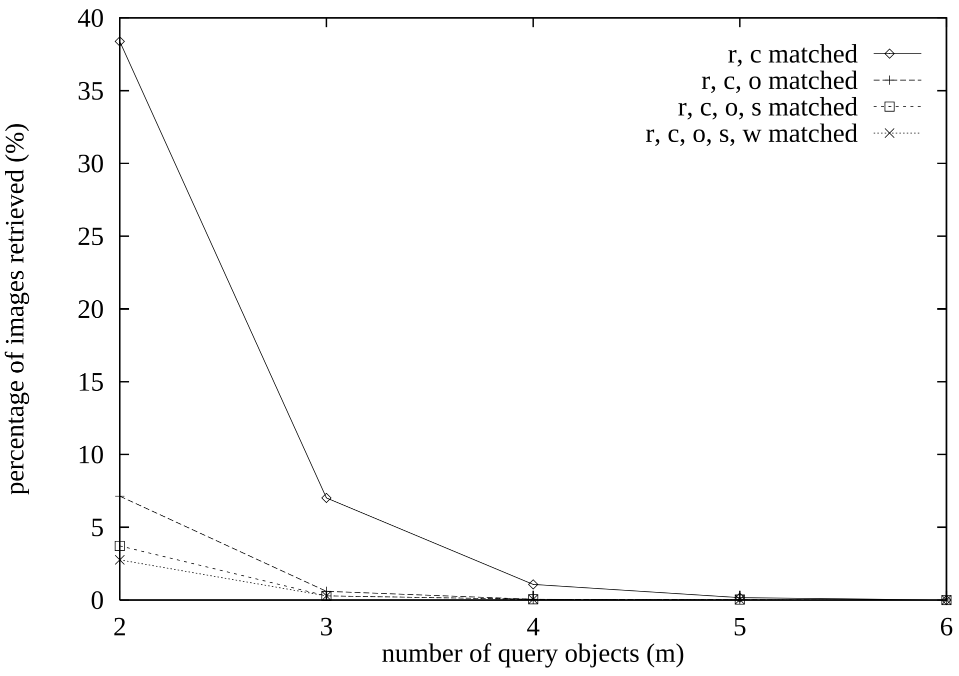


Figure 10: Medical IDB: average size of answer set, as a percentage of images retrieved, plotted against the number of query objects.

of the biggest object in the image. The quantization parameter q was set to 3 for all object properties.

Measurements of both the size of the answer sets and of the retrieval response times have been obtained and shown in Figures 10 and 11 respectively. To achieve average performance measures, for each value of m ranging from 2 to 6, 16 of the most common types of image queries (similar to the ones described in Section 6.1) have been applied and the average performance to queries specifying an equal number of objects has been computed. As expected, both the size the answer sets and of the retrieval response times decrease with the number of query objects. The size of the final answer set corresponding to queries specifying more than 3 objects is approximately 0%. Due to the small size of the IDB, retrievals are very fast and the overhead encountered in cases of indirect access queries results in retrievals which are, on the average, much slower than those shown in Figure 11. However, based on Figure 10, we may choose $K_{max} = 4$, since the size of the answer set corresponding to 4 objects is zero (in the average).

For $K_{max} = 4$, there are $10{,}760$ image subsets stored. Assuming $b_1 = 15$ bytes, the storage space per image is 714 bytes on the average. The primary attributes correspond to the relative positions and the size properties of objects (see Section 4.2). The time response to queries specifying less than 4 objects can become faster, when more than 2 primary attributes are used. However, the amounts of data which are retrieved and processed in response to queries specifying more than 4 objects are very small and retrievals cannot be made more efficient.

6.1 Examples of Retrievals

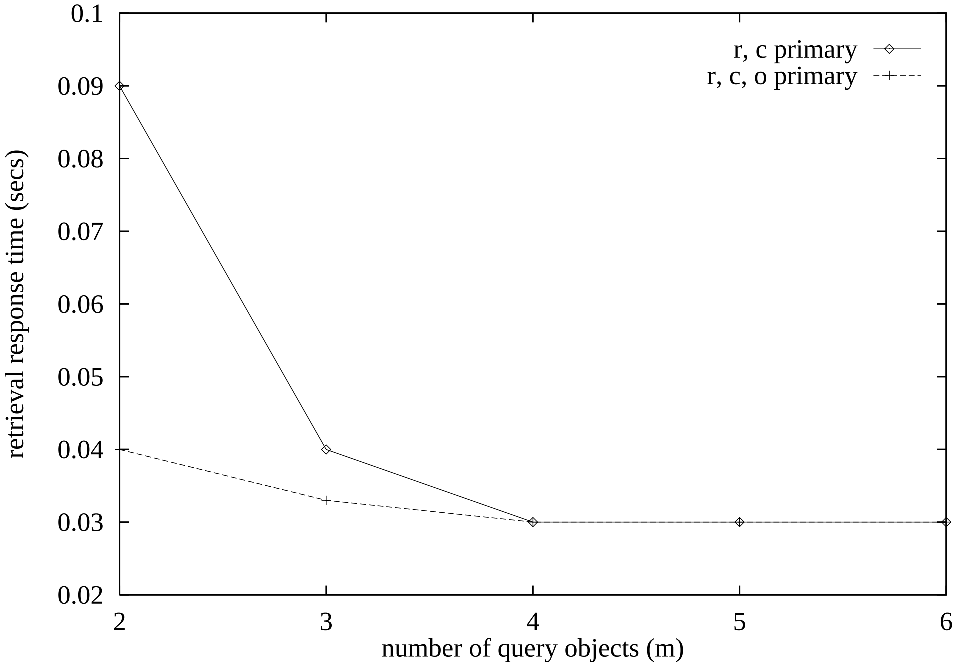
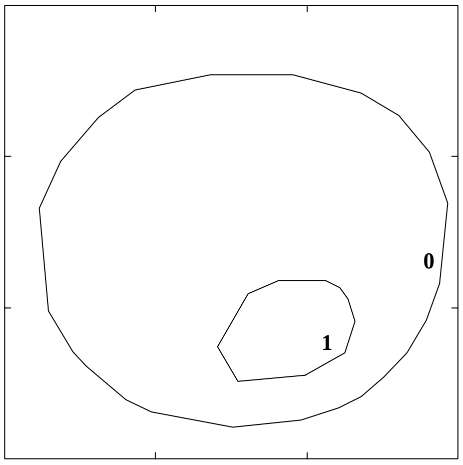


Figure 11: Medical IDB: average retrieval response times plotted against the number of query objects.

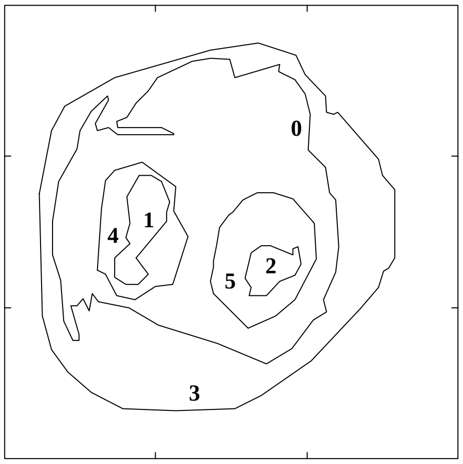
In the following, four characteristic examples of retrievals are presented and discussed. In Figure 12, query object 0 matches object 3 of the retrieved image and query object 1 matches object 2. Query and retrieved objects match each other with respect to their global characteristics (i.e., relative size, roundness and orientation) and have similar relative positions. However, they have rather different shapes, since objects have not been compared on the basis of local properties of their shape. Object 5 of the retrieved image matches query object 1 with respect to the properties of size and roundness but not with respect to orientation (object 5 has orientation 1, while query object 1 has orientation 0). Therefore, the answer does not include object 5.

In Figure 13 the image subsets matching the query are (3 2 6) and (3 5 6). Query object 0 matches both objects 2 and 5. Query object 1 is bigger than query object 2. Query object 1 matches object 3 of the retrieved image which is smaller than object 6 which matches query object 2. All these objects have size 0 with respect to the size to their outer object. To make similarity of sizes more accurate, quantization parameter q must take a value greater than 3.

The query of Figure 14 looks similar to the query of the previous example. However, object 1 is below object 0 (i.e., the center of mass of object 1 is below the center of mass of object 0), while in the previous example, object 1 was above object 0. The answer set is now completely different. In this case, we are faced with a problem of stability of representation (see Section 3.4). We can avoid this problem by applying the same query twice, once with rank string (0 1 2) and once with rank string (1 0 2). When the second rank string is used, the query has exactly the same representation with the query of Figure 13. Then, the image subset (6 0 2) will be included in the answer set. If the second ordering criterion were used, both query objects 0 and 1 would have assigned equal ranks. However, the second ordering criterion can



Query Image:
p = 0 1
r = 1 0
c = 2 1
s = 2 0
o = 0 0
w = 0 0



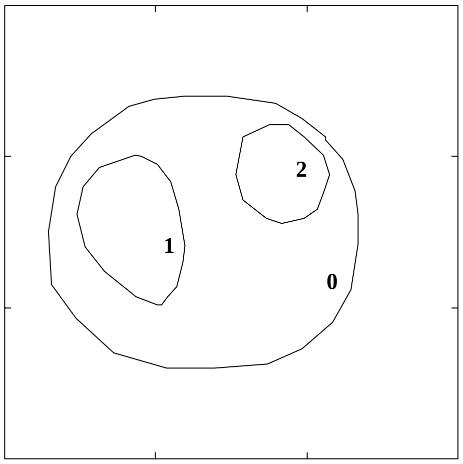
Retrieved Image:
Matching Subset: 3 2

Figure 12: Retrieval example 1.

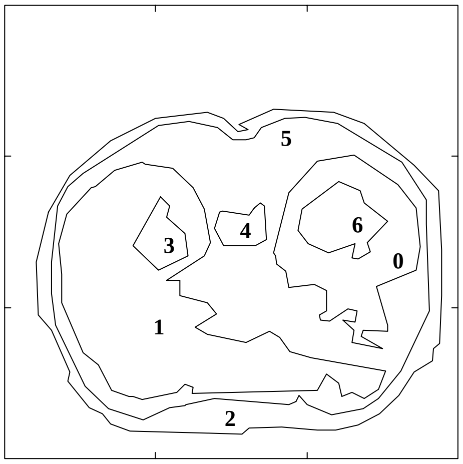
be used only in cases where images are not scaled with respect to each other.

The query of Figure 15 returns an empty answer set. The two images shown below the query image do not satisfy query criteria. However, there are cases in which we would like the left image to be included in the answer set, since both image subsets (7 2 5 0) and (7 2 5 1) match the query, except of object 7 which does not have the same orientation with query object 3. The left image is included in the answer set if the orientation is not used in comparisons. However, if the property of orientation is not used, then the right image and its subset (0 6 3 2) is included in the answer set, although it looks completely different than the query. To prevent images having different orientations (e.g., the right image) to be included in the answer set, a preprocessing step is required (before images are entered in the IDB) which puts all images in a standard orientation.

To deal with situations in which images match the query with respect to less than the total number of attributes, retrievals have to be performed in stages. Instead of always using the same set of attributes in image comparisons, at each stage of query processing, the user is allowed to choose the number and the kind of properties to be used in retrievals. There may be more than one answer sets returned in response to a given query (one at each stage of retrieval), while a query answer set may be enlarged due to inexact matching. To deal with such situations, an



Query Image:
p = 1 0 2
r = 1 0 2
c = 1 2 2
s = 0 2 0
o = 1 0 2
w = 1 1 1



Retrieved Image:
Matching Subset: 3 2 6
Matching Subset: 3 5 6

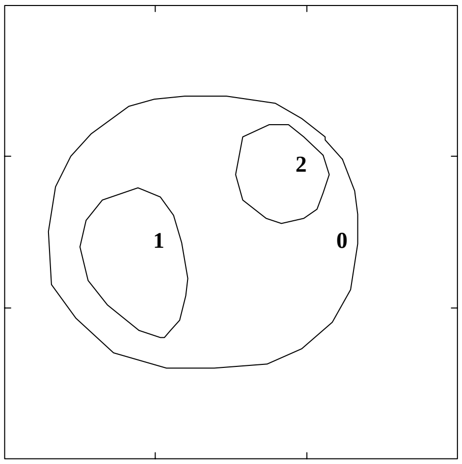
Figure 13: Retrieval example 2.

appropriate user interface, which supports this kind of query processing assisted by an efficient mechanism for browsing must be developed.

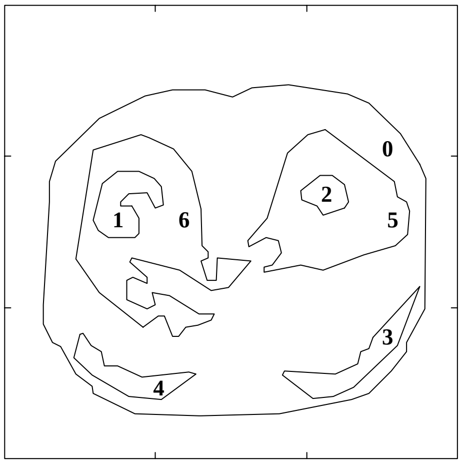
6.2 Accuracy of Retrievals

The evaluation of the accuracy of retrievals is subject to human interpretation and has to do with how good the methodology succeeds in retrieving images which a user expects to be returned in response to a given query. Retrievals are accurate, in the sense that only images satisfying query criteria are retrieved. However, in certain cases, such as those discussed in the previous section, the methodology may fail to retrieve images which look similar to the query.

Accuracy, is mainly a mater of image content representation and depends on how good a representation captures image content. The representations used so far are discrete (symbolic) representations of image content. However, in certain cases, these representations may be proven to be neither stable (see Section 3.4) nor complete (see Section 3.5). Increasing the accuracy of image content representations in IDB systems has become subject of independent research activities [27, 28]. Future proposals regarding image content representation, indexing and retrieval must take such efforts seriously into account.



Query Image:
p = 1 0 2
r = 0 1 2
c = 1 2 2
s = 0 2 0
o = 1 0 2
w = 1 1 1



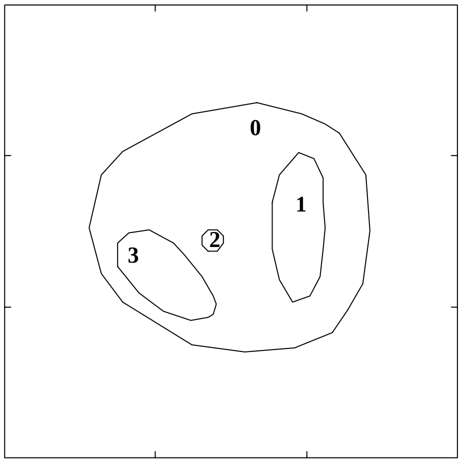
Retrieved Image:
Matching Subset: 6 0 2

Figure 14: Retrieval example 3.

7 Conclusions

In this paper, the requirements for the design and implementation of an IDB system, which supports the retrieval of images by content have been discussed and the characteristics of a prototype IDB system for medical images which is under development in our laboratory have been presented. This prototype system consists of a low-level subsystem for the storage of image data, tools and mechanisms for defining and creating the database model (schema) and an interactive user interface environment which is built on top of the storage subsystem. This environment provides a query language capable of manipulating various kinds of image data in image queries, and graphical tools facilitating the interaction between the user and the various system components. For example, in order to deal with the problems of efficiency, uncertainty, and complexity in describing the content of images, the user is allowed to interact with the IDB and edit the results of automated image segmentation. In addition, the user is allowed to specify the class to which an image belongs. Once the content of images has been extracted reliably, the system resumes responsibility for the efficient archiving and retrieval of images by content.

Attention is focused on a specific methodology which provides mechanisms for the efficient archiving and retrieval of images. In particular, a new methodology has been presented which



Query Image:
p = 3 2 0 1
r = 0 1 2 3
c = 1 2 2 0
s = 0 0 2 0
o = 2 1 2 1
w = 2 2 2 2

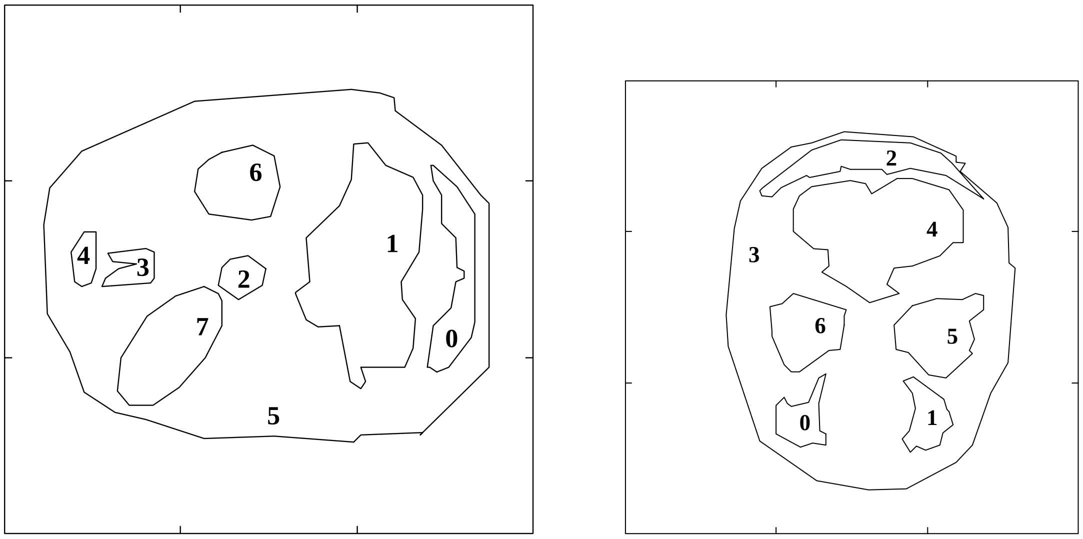


Figure 15: Retrieval example 4.

supports the efficient representation, indexing, and retrieval of images by content. Based on the known representations of 2-D strings, an indexing scheme has been proposed, which in contrast to 2-D strings, avoids an exhaustive search through the entire IDB. The representation of 2-D strings has been extended to take into account the inclusion relationships between objects and more than one object properties. In addition, two ordering criteria have been proposed and the representation of 2-D strings has been specialized to the cases of scaled and unscaled images respectively. Furthermore, the proposed methodology generalizes and extends the work of [12], which focuses entirely on the problem of indexing and is based on image subsets of size 2. In [12], problems relating to image content representation, retrieval methodologies and evaluation of system performance are not taken into account.

The proposed methodology ensures fast retrieval response times in cases of direct access queries. Queries are fast even in the case of indirect access queries, especially when K_{max} takes relatively large values (e.g., $K_{max} > 4$). The methodology is applicable in cases of images consisting of non-overlapping objects. However, even in cases where objects are partially visible, object positions may be specified by an expert user and indexing can still be based on the proposed representations. To ensure that all retrieved images satisfy the query criteria, one more step is needed in the search and retrieval process: all retrieved images must be compared to the query image on the basis of additional image properties and a more complete description of image content. 2-D C strings [28] provide a representation which has been shown to be better suited for the representation of images consisting of overlapping objects with complex shapes. However, any attempt at increasing the accuracy and specificity of retrieval by storing more complete descriptions of image content would result in lower response times and more demanding average requirements. This is a tradeoff which must be carefully considered in the context of specific applications.

The proposed methodology can be extended in many ways. One may consider the possibility of integrating, within the same indexing mechanism, additional kinds of image properties, either relational such as “connected to”, “overlap” etc., or properties specific to the shape of objects. Furthermore, one may consider the possibility of using higher level properties (e.g., properties defined in terms of others) or classes of images sharing common characteristics, thus creating a higher level of indexing and retrieval. The proposed methodology can also be extended to include the indexing of image sequences in three or four dimensions by ordering objects in a space of three or four dimensions respectively, where the fourth dimension is time.

The retrieval capabilities of an IDB system can be significantly enhanced by extending the user interface with additional tools and mechanisms for image processing and with a powerful query language supporting the treatment of variable similarity criteria as well as the processing of various types of image queries. In particular, the methodology may be extended to support the processing of fuzzy queries e.g., “partial match” and “range” queries [29, 30]. In partial match queries, one or more properties of objects contained in a query image are left unspecified, in which case, these properties are allowed to take any value in a specific domain. In range queries, instead of exact values, ranges of values of one or more properties corresponding to one or more query objects are specified.

Acknowledgement

This work was supported in part by projects HIPACS and EURIPACS, under programme AIM of the EEC.

References

[1] Won Kim. Object Oriented Databases: Definitions and Research Directions. *IEEE Transactions on Knowledge and Data Engineering*, 2(3):327–341, September 1990.

[2] John V. Joseph, Satish M. Thatte, Craig W. Thompson, and David L. Wells. Object Oriented Databases: Design and Implementation. *Proceedings of the IEEE*, 79(1):42–64, January 1990.

[3] Shi-Kuo Chang. *Principles of Pictorial Information Systems Design*, chapter 8, pages 172–211. Prentice Hall International Editions, 1989.

[4] Makoto Nagao. Control Strategies in Pattern Analysis. *Pattern Recognition*, 17(1):45–56, 1984.

[5] King-Sun Fu and Azriel Rosenfeld. Pattern Recognition and Computer Vision. *IEEE Computer*, 17(10):274–282, 1984.

[6] Christos Faloutsos and Yi Rong. Spatial Access Methods Using Fractals: Algorithms and Performance Evaluation. Technical Report UMIACS-TR-89-31, CS-TR-2214, University of Maryland, Colledge Park, Maryland, February 1989.

[7] Christos Faloutsos. Access Methods for Text. *ACM Computing Surveys*, 17(1):49–74, March 1985.

[8] Panos Constantopoulos, Stelios C. Orphanoudakis, and Euripides G. Petrakis. An Approach to Multimedia Document Retrieval on the Basis of Pictorial Content. Technical Report 011, Institute of Computer Science, Foundation of Research and Technology - Hellas, Heraklion, Greece, February 1988.

[9] Martin A. Fischler and Robert A. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, c-22(1):67–92, 1973.

[10] Linda G. Shapiro and Robert M. Haralick. Structural Discriptions and Inexact Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):504–519, 1981.

[11] Shi-Kuo Chang, Qing-Yun Shi, and Cheng-Wen Yan. Iconic Indexing by 2-D Strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):413–428, May 1987.

[12] Chin-Chen Chang and Suh-Yin Lee. Retrieval of Similar Pictures on Pictorial Databases. *Pattern Recognition*, 24(7):675–680, 1991.

[13] Petros Kofakis and Stelios C. Orphanoudakis. Graphical Tools and Retrieval Strategies for Image Indexing by Content. In *Proceedings of Computer Assisted Radiology, CAR91*, pages 519–524, Berlin, July 1991.

[14] Petros Kofakis and Stelios C. Orphanoudakis. Image Indexing by Content. In M. Osteaux et. al., editor, *A Second Generation PACS Concept*, chapter 7, pages 250–293. Springer-Verlag, 1992.

[15] Euripides G.M. Petrakis and Stelios C. Orphanoudakis. Tools and Methodologies for the Indexing, Storage and Retrieval of Medical Images. In A. Todd-Pokropek and H. Lemke, editors, *Proceedings of Computer Assisted Management of Medical Images*, London, UK, 1992.

[16] Jayaram K. Udupa, Hsiu-Mei Hung, Dewey Odhner, and Roberto Goncalves. Multidimensional Data Format Specification: A Generalization of the American College of Radiology - National Electric Manufacturers Association Standards. *Journal of Digital Imaging*, 5(1):26–45, February 1992.

[17] M. Carey, D. DeWitt, and E. Shekita. Storage Management for Objects in Exodus. In W. Kim and F. Lochovsky, editors, *Object-Oriented Concepts, Databases and Applications*. Addison-Wesley, 1989.

[18] Alexandros Billiris. The performance of Three Database Storage Structures for Managing Large Objects. In *Proceedings of the 1992 ACM SIGMOD*, pages 276–285, San Diego, California, June 1992.

[19] Dennis Stanton and Dennis White. *Constructive Combinatorics*, chapter 1, pages 1–25. Springer-Verlag, 1986.

[20] Edward M. Reingold, Jurg Nievergelt, and Narsingh Deo. *Combinatorial Algorithms*, chapter 5, pages 90–91. Prentice Hall, 1977.

[21] Theo Pavlidis. *Algorithms for Graphics and Image Processing*, chapter 15, pages 316–357. Computer Science Press, 1981.

[22] Euripides G.M. Petrakis. *Image Representation, Indexing and Retrieval Based on Spatial Relationships and Properties of Objects*. PhD thesis, University of Crete, Department of Computer Science, March 1993. Available as Technical Report FORTH-ICS/TR-075.

[23] Berthold Klaus Paul Horn. *Robot Vision*, chapter 3, pages 46–64. MIT Press, 1986.

[24] Wen-Hsiang Tsai and Shiaw-Shian Yu. Attributed String Matching with Merging for Shape Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):453–462, 1985.

[25] Alan Kalvin, Edith Schonberg, Jacob T. Schwartz, and Micha Sharir. Two-Dimensional, Model-Based, Boundary Matching Using Footprints. *The International Journal of Robotics Research*, 5(4):38–55, 1986.

[26] Fridtjof Stein and Gerard Medioni. Efficient Two Dimensional Object Recognition. In *In Proccedings of 10th International Conference On Pattern Recognition*, pages 13–17, Atlantic City, New Jersey, June 1990.

[27] Erland Jungert and Shi-Kuo Chang. An Algebra for Symbolic Image Manipulation and Transformation. In T. L. Kunii, editor, *Visual Database Systems*. Elsevier Science Publishers B.V. (North Holland), 1989.

[28] Suh-Yin Lee and Fang-Jung Hsu. 2D C-String: A New Spatial Knowledge Representation for Image Database Systems. *Pattern Recognition*, 23(10):1077–1087, 1990.

[29] Christos Faloutsos. Gray Codes for Partial Match and Range Queries. *IEEE Transactions on Software Engineering*, 14(10):1381–1393, October 1988.

[30] Christos Faloutsos and Shari Roseman. Fractals for Secondary Key Retrieval. Technical Report UMIACS-TR-89-47, CS-TR-2242, University of Maryland, Colledge Park, Maryland, May 1989.