Similarity Searching in Large Image DataBases

Euripides G.M. Petrakis[*]
MUltimedia Systems Institute of Crete (MUSIC),
Technical University of Crete,
PO BOX 134 Chania, GR 73100 Greece.
e-mail: petrakis@ced.tuc.gr

Christos Faloutsos[†]
Institute for Systems Research (ISR) and
Department of Computer Science
University of Maryland.
e-mail: christos@cs.umd.edu

Abstract

We propose a method to handle approximate searching by image content in large image databases. Image content is represented by attributed relational graphs holding features of objects and relationships between objects. The method relies on the assumption that a fixed number of "labeled" or "expected" objects (e.g., "heart", "lungs" etc.) are common in all images of a given application domain in addition to a variable number of "unexpected" or "unlabeled" objects (e.g., "tumor", "hematoma" etc.). The method can answer queries by example such as *"find all X-rays that are similar to Smith's X-ray"*. The stored images are mapped to points in a multidimensional space and are indexed using state-of-the-art database methods (R-trees). The proposed method has several desirable properties: (a) Database search is approximate so that all images up to a pre-specified degree of similarity (tolerance) are retrieved, (b) it has no "false dismissals" (i.e., all images qualifying query selection criteria are retrieved) and (c) it scales-up well as the database grows. We implemented the method and ran experiments on a database of synthetic (but realistic) medical images. The experiments showed that our method significantly outperforms sequential scanning by up to an order of magnitude.

Index Terms: image database, image retrieval by content, query by example, image content representation, attributed relational graph, image indexing, R-tree, similarity searching.

1 Introduction

In many applications, images comprise the vast majority of acquired and processed data. In remote sensing and astronomy, large amounts of image data are received by land stations for processing,

[*]Please address all correspondence to this author. This research was partially funded by project DAIDALOS, ECUS008:9828, EC-US Exploratory Activity and partially by project MILORD A2024, under programme AIM of the EEC.

[†]His research was partially funded by the National Science Foundation under Grants IRI-9205273 and IRI-8958546 (PYI), with matching funds from EMPRESS Software Inc. and Thinking Machines Inc.

analysis and archiving. A similar need of processing, analysis and archiving of images has been identified in applications such as cartography (images are analog or digitized maps) and meteorology (images are meteorological maps). In medicine, in particular, a large number of images of various imaging modalities (e.g., Computer Tomography, Magnetic Resonance etc.) are produced daily and used to support clinical decision making. The capabilities of the above application fields can be extended to provide valuable teaching, training and enhanced image interpretation support, by developing techniques supporting the automated archiving and the retrieval of images by content. For example, in medicine, before making a diagnosis, a clinician could retrieve similar cases from the medical archive. Content-based retrievals would not only yield cases of patients with similar image examinations and similar diagnosis but also, cases of patients with similar image examinations and different diagnoses [42].

To support queries by image content in an Image DataBase (IDB), all images must be analyzed prior to storage so that, descriptions of their content can be extracted and stored in the database together with the original images. These descriptions are then used to search the IDB and to determine which images satisfy the query selection criteria. The effectiveness of an IDB system ultimately depends on the types and correctness of image content representations used, the types of image queries allowed and the efficiency of search techniques implemented.

Fast responses are essential to an IDB. An IDB system must employee searching methods that are faster than sequential scanning methods, and which must "scale-up" well (i.e., their performance remains consistently better than the performance of sequential scanning methods as the database grows).

Query formulation must be flexible and convenient (as opposed to queries expressed by a command-oriented query language like SQL). Ideally, queries must be specified through a graphical user interface, such as by example (i.e., by providing an example image or by drawing a sketch on the screen). Query by example permits even complicated queries: The user may specify several objects with complex shapes and inter-relationships and may ask for all images containing similar objects with similar relationships. The retrieved images need not be exactly similar to the query. Instead, database search must be approximate so that, all images up to a pre-specified degree of similarity (tolerance) are retrieved.

In this work we deal with the following problem: Given a set of images, retrieve those which are similar to an example query (e.g., "find all X-rays that are similar to Smith's X-ray"). We propose a general methodology which (a) uses an efficient representation of image content based on "Attributed Relational Graphs" (ARGs), (b) indexes the stored ARGs with state-of-the-art database methods (R-trees), and (c) supports approximate retrieval of images by content (i.e., based on both object properties and relationships between objects).

We design the image distance/similarity functions and we show that the search method allows no “false dismissals” (i.e., all images qualifying similarity criteria are retrieved). Specifically, all images within a given tolerance are retrieved. The performance of the proposed methodology has been evaluated based on an IDB of synthetic, but realistic, medical images. The results of this evaluation demonstrate very significant performance improvements over traditional sequential scan techniques utilizing graph matching. Finally, we show that the method scales-up well.

The rest of this paper is organized as follows: The definition of the problem, the assumptions made and a short presentation of the underlying theory are presented in Section 2. A review of related work done in the areas of Computer Vision and DataBases is presented in Section 3. The proposed methodology is presented in Section 4. In Section 5, experimental results are given and discussed. In section 6, we make several interesting observations on the proposed approach and we discuss optimization techniques along with issues for future research. Finally, the conclusions are given in Section 7.

2 Problem Definition and Background

Given a collection of N images, we must derive appropriate representations of their content and organize the images together with their representations in the IDB so that we can search efficiently for images similar to an example image.

All images are segmented into close contours corresponding to dominant image objects or regions. We assume that all images contain a number of “expected” or “labeled” objects. These are objects common in all images of a given application domain. For example, in medical images, the expected objects may correspond to the usual anatomical structures (e.g., “heart”, “lungs”) and the outline contour of the body. Similarly, the labeled objects may be the cell outline in microscope images; the sun and the horizon in outdoor images etc. All expected objects are identified prior to storage and a class or name is assigned to each one. The labeled objects need not be similar in all images.

Not all objects need to be identified: Images may also contain “unexpected” or “unlabeled” objects. These, may be either objects not present in all images or objects whose characterization is difficult or ambiguous. For example, in medical images, the unexpected objects may correspond to abnormal (pathological) structures (e.g., “hematoma”, “tumor” etc.).

In this work, we deal with images containing a fixed number (k) of labeled objects and a variable number ($u \geq 0$) of unlabeled objects. We also assume that the labeled objects have different labels. Objects not common in all images are treated as unlabeled (unexpected). Similarly, queries by example may specify a fixed number of labeled objects and a variable number of unlabeled objects. Notice that this is a general setting. One special case is the case where all objects are unlabeled ($k = 0$). Another special case is when all objects are labeled ($u = 0$) in all images as was the case in [2]. There, the problem was to search a database of face images; from each image, a fixed number of labeled objects are identified (eyes, nose, etc.) and their attributes and relative positions are computed.

2.1 Background

Image descriptions are given in terms of object properties and in terms of relationships between objects. Such image descriptions are represented by relational structures such as Attributed Relational Graphs (ARGs) [4, 11]. In an ARG, the objects are represented by graph nodes and the relationships between objects are represented by arcs between such nodes. Both nodes and arcs are labeled by attributes corresponding to properties (features) of objects and relationships respectively.

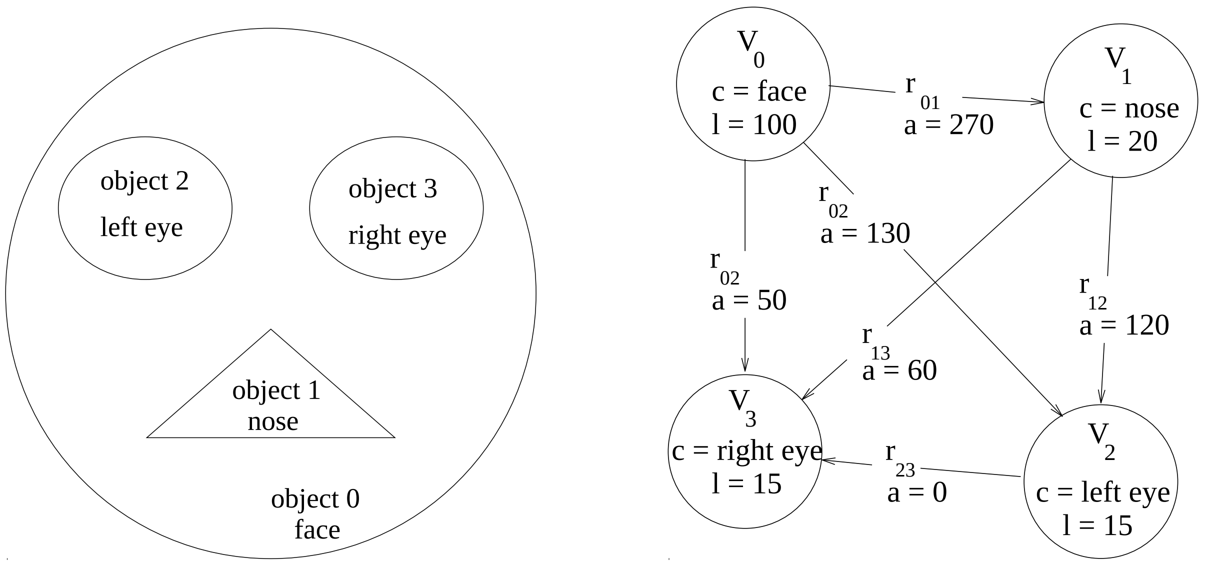

Figure 1: *Example image showing a sketch of a face (left) and its corresponding ARG (right).*

Figure 1 shows an example image (a line drawing showing a face) containing four objects (numbered 0 through 3) and its corresponding ARG. Each object has an attribute (c) denoting its name or class and an attribute representing the length (l) of its boundary. The relationship between any two objects has also one attribute, the angle (a) with the horizontal direction of the line connecting the centers of mass of these objects.

The specific features which are used in ARGs are derived from the raw image data and, depending on the application, can be geometric (i.e., independent of pixel values), statistical or textural, or features specified in some transform domain (e.g., Fourier coefficients of object shapes). In the case of medical CT and MRI images used in this work, the set of features is given in Section 4.1. However, the proposed methodology is independent of any specific kind of features.

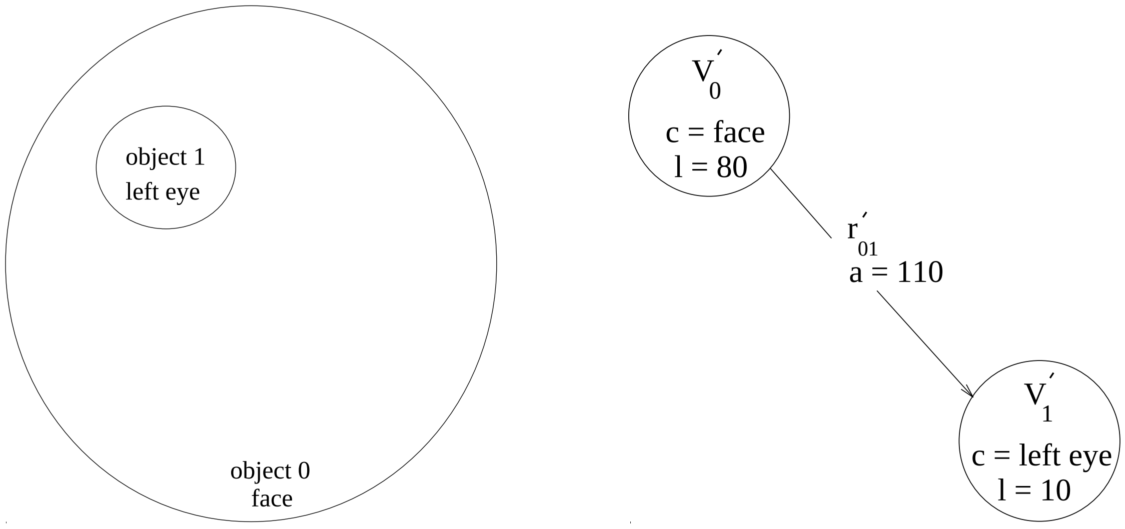

Figure 2: *Example query image (left) and its corresponding ARG (right).*

The problem of retrieving images which are similar to a given example image is transformed into a problem of searching a database of stored ARGs: Given a query, its ARG has to be computed and compared with all stored ARGs. Matching between ARGs is a well known problem and has been studied extensively in the Computer Vision literature [23, 54, 4, 18]. Specifically, matching a query and a stored graph is treated as a subgraph isomorphism problem.

Figure 2 shows an example query and its corresponding ARG. In this example, query object 0 can only be associated with object 0 of the image of Figure 1 since, this is the only object having the same label with it. Similarly, query object 1 is matched with object 2. Their corresponding relationships are matched too. Equivalently, query node v'_0 is associated to node v_0, v'_1 to v_2 and arc r'_{01} is associated to arc r_{02} of the graph of the original image. However, if the label of a query object is unknown, all possible associations between this query object and the objects in the original image have to be examined. The problem becomes harder if the query or the original image contain many unlabeled objects or objects with the same label. Then matching becomes a hard combinatorial problem.

In comparisons between ARGs, we need a measure of the "goodness" of matching. A measure of goodness is defined in [4]: Let Q be a query image consisting of q objects and S be a stored image consisting of s objects. Let $F()$ be a mapping from objects in Q to objects in S (e.g., such a mapping associates objects with the same labels). The cost of this mapping is defined as:

$$Dist_F(Q,S) = \sum_{i \in [1,q]} COST\,(i, F(i)) + \sum_{i,j \in [1,q]} COST\,(i, j, F(i), F(j)) \tag{1}$$

The first term in Equation 1 is the cost of matching associated nodes, while the second term is the cost of matching the relationships between such nodes.

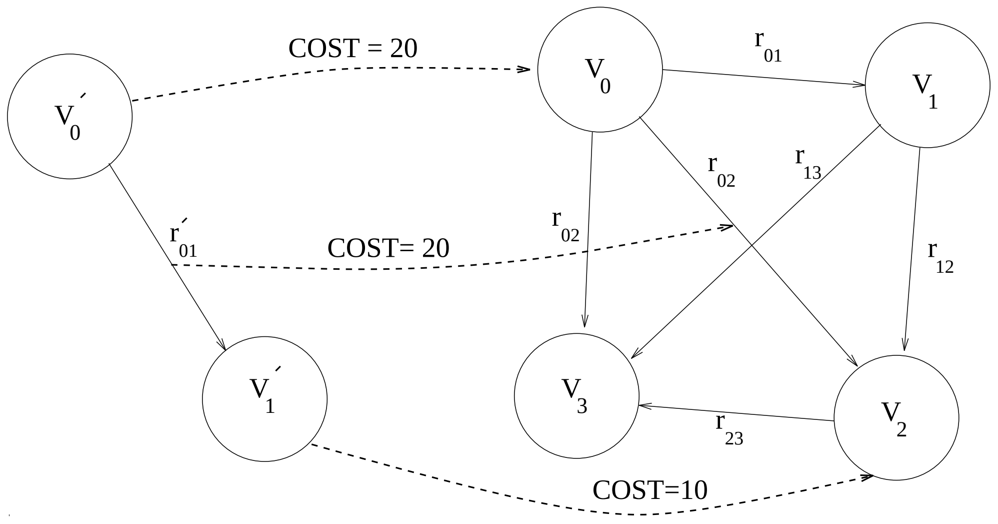

Figure 3: *Matching between the query and the original example image.*

In our setting, only a subset of the objects in the stored image S need to be matched. There is no cost if the data image contains extra objects; however, we assume that the cost is infinite if the data image is missing one of the objects of the query.

$COST$ is the cost of matching features of objects or features of relationships between associated objects. The distance between images Q and S is defined as the minimum distance computed over all possible mappings $F()$:

$$Dist(Q,S) = \min_F \{Dist_F(Q,S)\}. \tag{2}$$

The typical way to compute $Dist_F(Q,S)$ is using an L_p metric. This is done as follows: Let $(q_1, q_2 \ldots q_K)$ be a vector of feature values derived from Q by taking the features of all its objects and of their relationships in some pre-specified order (e.g., object 1 and its relationships with the remaining objects are taken first followed by the features of object 2 etc.). Let $(s_1, s_2, \ldots s_K)$ be the vector derived from S by taking the features of the objects associated to objects in Q in the same order. Then, Equation 1 can be written as follows:

$$Dist_F(Q,S) = Dist_{p,F}(Q,S) = \left[\sum_{i=1}^{K} |q_i - s_i|^p\right]^{1/p} \tag{3}$$

p is the order of the metric. For $p = 1$ and $p = 2$ we obtain the Manhattan (city-block) and the Euclidean distance respectively. For example, the Manhattan distance between the query image of Figure 2 and the example image of Figure 2 is $Dist(Q, S) = |100 - 80| + |15 - 10| + |130 - 110| = 45$. We have omitted the subscript F because there is only one mapping.

Similarity searching in an IDB of stored ARGs requires that all images within distance t must be retrieved. Specifically, all images qualifying the following condition have to be retrieved:

$$Dist(Q, S) \leq t. \tag{4}$$

Without loss of generality, we use the Euclidean distance ($p = 2$). However, the proposed method can handle *any* L_p metric.

3 Survey - Related Work

Important considerations in the design and implementation of IDB systems supporting queries by image content are: Image feature extraction, image content representation and organization of stored information, search and retrieval strategies, and user interface design. Addressing such issues has become object of intensive research activities in many areas of Computer Science over the past few years [57, 11, 35, 13]. Advances mainly in the areas of Databases and Computer Vision research resulted in methods which can be used for image archiving, retrieval and IDB design work. However, as observed in [32], there is a need for increased communication between the vision and the database communities to deal with the above issues. Combining results from both areas is an important next step.

3.1 Image Retrieval by Content

Image content can be described indirectly through attributes (e.g., subject, speaker, etc.) or text (e.g., captions) [16]. However, queries by image content require that, prior to storage, images are processed, appropriate descriptions of their content are extracted, stored in the database and used in retrievals.

Retrievals by image content is not an exact process (two images are rarely identical). Instead, all images with up to a pre-specified degree of similarity have to be retrieved. The design of appropriate image similarity/distance functions is a key issue here. In addition, retrievals can be greatly accelerated if the stored images are indexed. So far, most of the methods which have been developed, perform either exact match retrievals (e.g., 2-D strings [15]) or exhaustive (sequential) database search (e.g., [9, 40, 19]).

Approaches to combine approximate database search and indexing do exist [31, 20]. However, such techniques do not support image retrievals by content (i.e., based on properties of objects contained in images and on relationships between objects). The proposed methodology achieves this goal by using ARG representations of image content in combination with state-of-the-art database techniques for indexing in many dimensions, called “spatial access methods”.

3.1.1 Exact Match Searching in Image Databases

Once an image description has been derived, it can be easily represented in database storage structures (e.g., database relations) [12, 43]. Such image representations can be used to answer queries specifying simple constraints on object property values or relationships. Queries by example are difficult to be processed.

2-D strings [15] provide an approach to efficient image content representation and reduced complexity (i.e., polynomial) matching in image databases. 2-D strings assume that image objects are identified prior to storage so that a unique name or class is assigned to each one. The relative positions between all objects are then represented by two one dimensional strings. The idea is to project object positions (e.g., their centers of mass) on the x and y axis respectively, and take the objects in the same order as they appear in the two projections. The problem of image retrieval is then transformed into one of string matching: All 2-D strings containing the 2-D string of the query as a substring, are considered similar to it and are retrieved. To speedup retrievals, methods for indexing 2-D strings in a database has been proposed in [10, 46, 45]. Extensions of 2-D strings to treat various types of image properties has been proposed in [46]. Representations such as 2-D G strings [14] and 2-D C strings [36], have also been proposed and deal with situations of overlapping objects with complex shapes. However, such representations are not as simple and compact as the original 2-D strings.

The effectiveness of 2-D string based representations in retrieving images by content has been investigated in [46]. It has been shown that, with respect to what a user expects to see in the responses, 2-D strings may yield “false alarms” (not qualifying images) and “false dismissals” (qualifying but not retrieved images). This is mostly due to the fact that, 2-D strings change drastically with small variations in object characteristics. Techniques for inexact match retrievals based on 2-D string has also been proposed in [38, 37]. However, such retrievals are less time efficient (i.e., matching is not polynomial).

3.1.2 Approximate Searching in Image Databases

A system designed to support the segmentation, the description, as well as the interactive retrieval of facial images from an IDB is presented in [2]. A-priori knowledge regarding the kind and the positioning of expected image objects (e.g., face outline, nose, eyes etc.) is employed and used to guide the segmentation of face images into disjoint regions corresponding to the above objects. Retrieval is performed in stages allowing the user to adjust the query criteria at each stage and progressively achieve the desired results. This strategy may be especially helpful when the specification of pictorial content in queries is ambiguous and it may be proven to be an effective method for making a difficult final selection. However, database search is exhaustive, and each time a new query is issued, the whole database has to be searched. This may be very time consuming for large databases. The authors do not provide experimental results on the time performance of their method. This work may benefit greatly from the proposed approach on image indexing and retrieval by content in two ways: (a) Database search can be greatly accelerated with our proposed indexing method and (b) queries may include also unlabeled or unexpected objects (e.g., a scar along the face).

In [47], an information retrieval approach is proposed. The user can specify "imprecise" queries (i.e., queries that do not evaluate into "yes" or "no") using a command oriented query language which allows the user to express preference and importance values for the objects involved in the query. Retrievals are based on a multilevel signature technique. The method assumes that the number and the kind of objects which appear in all images are known in advance. However, database search is exhaustive and the method cannot handle efficiently queries by example. The authors do not provide experimental results.

An attempt to combine indexing and approximate database search is described in [31]. The main idea is to extract f features from each image, thus mapping images into points in a f-dimensional space. Once this is achieved, any spatial access method can be used to handle range and nearest-neighbor queries efficiently. This approach has been applied for the recognition of written digits: The bitmap of each digit is represented by a few (3 to 5) rectilinear rectangles whose coordinates collectively are used as the features. The method did not address the issue of false dismissals, nor the problem of retrieving images by specifying properties of objects and relationships between objects.

In the QBIC project of IBM [20], an indexing method for queries on color, shape and texture is proposed. The main contribution in this paper was a technique to handle the cases where the distance function is not an Euclidean distance including "cross-talk" of attributes. Focusing mainly on colors, this work does not show how to handle multiple objects per image, as well as their inter-relationships. Thus, the paper does not show how to handle queries of the form “find the images where a red ball is close to a yellow rectangle”. Other methods supporting retrieval of images based on the colors in a scene are proposed in [8, 55, 34].

Additional work on image content retrieval includes: The technique of [56] deals with the problem of image content retrieval based on spatial relationships between objects when images are rotated. The encoding of relationships, called “arrangement”, is independent of image rotation, and describes the sequence in which the neighbors of each object are arranged around it. The work in [29] deals with the problem of indexing for medical images. Given a picture, the four “most important” objects are taken and their centers of mass are used to represent their spatial relationships. However, this approach seems to allow for false dismissals: If an X-ray has 5 objects, one of them will be ignored. Thus, queries on this fifth object will never retrieve that X-ray.

3.2 Spatial Access Methods

A tool to achieve faster-than-sequential searching is to use the so-called “spatial access methods”. These are file structures to manage a large collection of f-dimensional points (or rectangles or other geometric objects) stored on the disk so that, “range queries” can be efficiently answered. A range query specifies a region (e.g., hyper-rectangle or hyper-sphere) in the address space, requesting all the data objects that intersect it. If the data objects are points (as eventually happens in our application), the range query requires all the points that are inside the region of interest. An example of a range query on point data is “retrieve all the cities that are 200 km away from Brussels”.

Several spatial access methods have been proposed. A recent survey can be found in [51]. These methods can be grouped into the following classes: (a) Methods that transform rectangles into points in a higher dimensionality space [28]; (b) methods that use linear quad-trees [25, 1] or, equivalently, the “z-ordering” [41] or other “space filling curves” [22, 30]; and finally, (c) methods based on trees (k-d-trees [6, 7], k-d-B-trees [49], hB-trees [39], cell-trees [26] etc.). One of the most characteristic approaches in the last class is the R-tree [27].

The R-tree can be envisioned as an extension of the B-tree for multidimensional objects. A geometric object is represented by its Minimum Bounding Rectangle (MBR). Non-leaf nodes contain entries of the form (ptr, R) where ptr is a pointer to a child node in the R-tree; R is the MBR that covers all rectangles in the child node. Leaf nodes contain entries of the form $(object-id, R)$ where $object-id$ is a pointer to the object description, and R is the MBR of the object. The main idea in the R-tree is that father nodes are allowed to overlap. This way, the R-tree can guarantee good space utilization and remain balanced. Figure 4 illustrates data rectangles (in black) organized in

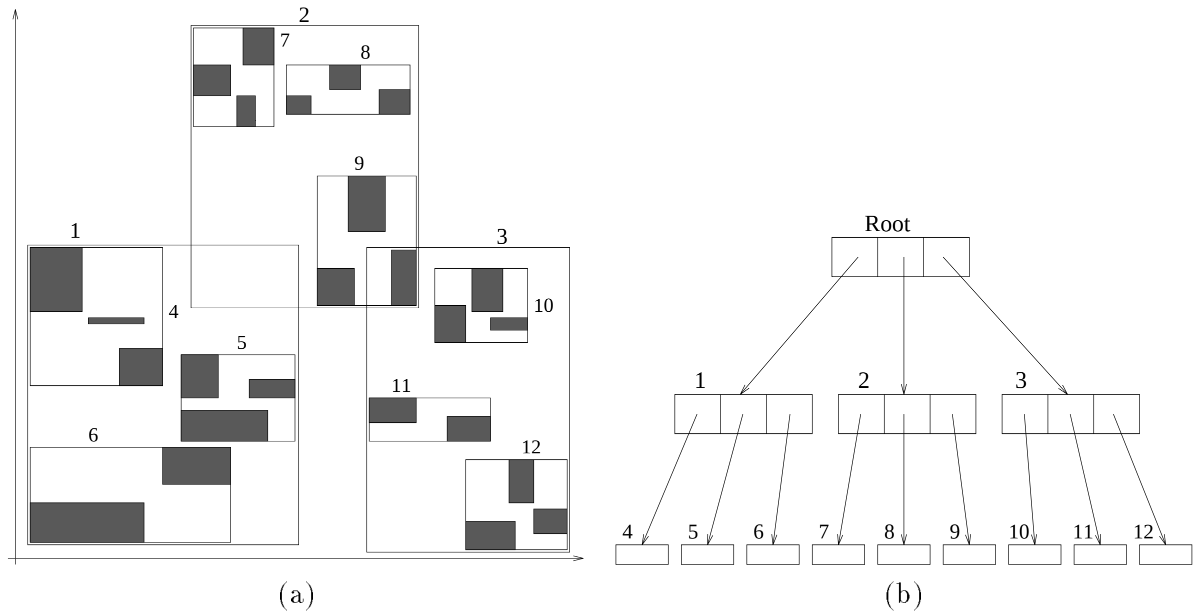

Figure 4: *Data (dark rectangles) organized in an R-tree (a), and the resulting tree on disk (b).*

an R-tree (left); the file structure for the same R-tree is also shown (right); the nodes correspond to disk pages. Extensions, variations and improvements to the original R-tree structure include the packed R-trees [50], the R^+-tree [53] and the R^*-tree [5].

4 Proposed Solution

In this work, we use the R-tree as the underlying method for indexing images by content. The reason for our choice is that the R-tree is more robust in high-dimensionality address spaces. As we shall see, in IDB applications the number of attributes/dimensions can be high (20-30), which can still be handled by an R-tree [44]. On the contrary, the main competitors of the R-tree (see Section 3.2) require time or space that is exponential on the dimensionality.

Henceforth, we assume that images are segmented into closed contours corresponding to labeled and unlabeled objects of interest. Figure 5 shows an example of an original MRI image and of its segmented form. We can easily identify three labeled objects namely, spine (object 2), liver (object 1) and body outline (object 0). Henceforth, the number of labeled objects in all images will be $k = 3$.

Techniques for the automatic segmentation and recognition of tomographic images do exist [3, 33, 17, 48]. In general, image automatic segmentation and labeling of the components are outside the scope of this paper. For our purposes, we assume that each image has been segmented automatically or manually and that its components have been labeled (typically, by a domain

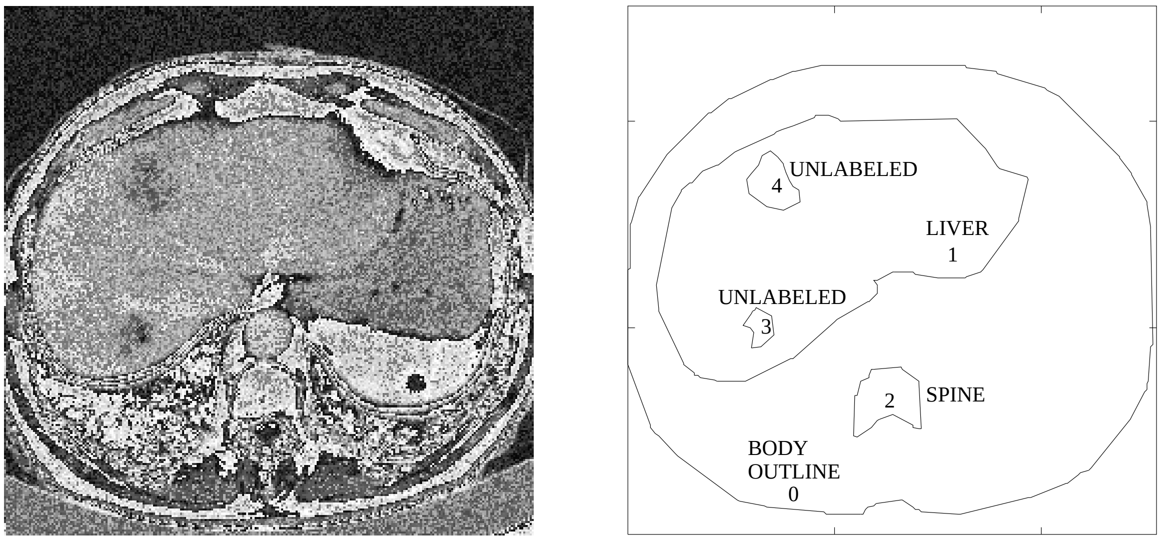

Figure 5: *Example of an original grey-level image (left) and its segmented form (right) showing 3 labeled objects (body outline with index 0, liver with index 1 and spine with index 2) and 2 unlabeled with indices 3 and 4.*

expert). The contribution of our work is on the fast searching after the images and the queries have been segmented and labeled.

4.1 Image Content Description

The descriptions used in this work are given in terms of properties of objects contained in images and in terms of relationships between such objects. Individual objects are described by properties corresponding to characteristics of their position, size and shape. Specifically, the following set of properties is used to describe each object:

- *Size (s)*, computed as the size of the area it occupies.
- *Roundness (r)*, computed as the ratio of the smallest to the largest second moment.
- *Orientation (o)*, defined to be the angle between the horizontal direction and the axis of elongation. This is the axis of least second moment.

The following properties are used to describe the spatial relationships between two objects:

- *Distance (d)*, computed as the minimum distance between all pairs of line segments, taking one from each object.

- *Relative Position (p)*, defined as the angle with the horizontal direction of the line connecting the centers of mass of the two objects.

The above set of features is by no means unique. Additional features that could be used include the average grey-level and texture values, moments or Fourier coefficients etc. as object descriptors; Relative size, amount of overlapping or adjacency etc. can be also used to characterize the relationships between objects. However, it has been shown in [43, 46] that the proposed set of features give acceptable results in retrievals of medical images by content. Notice that, the proposed method can handle any set of features that the domain expert may deem appropriate. The contribution of this work is not on choosing a good set of features, but on accelerating the sequential search on a given, expertly chosen, set of features.

Figure 6 shows the attributed graph representing the content of the example image of Figure 5. Nodes correspond to objects and arcs correspond to relationships between objects. Both nodes and arcs are labeled by attribute values corresponding to object properties and relationships respectively. The properties corresponding to the arc connecting object 1 (liver) with object 2 (spine) are not shown. Angles are in degrees.

The derived representation are both scale and translation invariant (i.e., images translated or scaled with respect to each other result in the same representation). To achieve scale invariance, we normalize lengths and areas by dividing them respectively by the diameter and the area of the largest object of the given image. The features we have chosen are also independent of image translation since, only relationships between objects are used to characterize object positions. To achieve rotation invariance, all images are registered to a standard orientation

4.2 Image Indexing - File Structure

Our goal is to achieve fast searching in a database of stored ARGs. The approach we follow is to map ARGs into points in a multidimensional space. Then, a multidimensional access method can be used. However, such a method requires that the number of dimensions (equivalently, the number of keys or axes) are known in advance and are fixed. However, the number of objects is not the same in all images, and therefore, the number of dimensions cannot be fixed. We solve this problem by decomposing each image into images containing an equal number of objects, called "sub-images":

Definition 1 A (k, u) sub-image of a given image contains the k labeled objects and u unlabeled objects.

An attribute vector is then computed to each derived sub-image consisting of property values corresponding to the objects it contains and the relationships between these objects taking them in

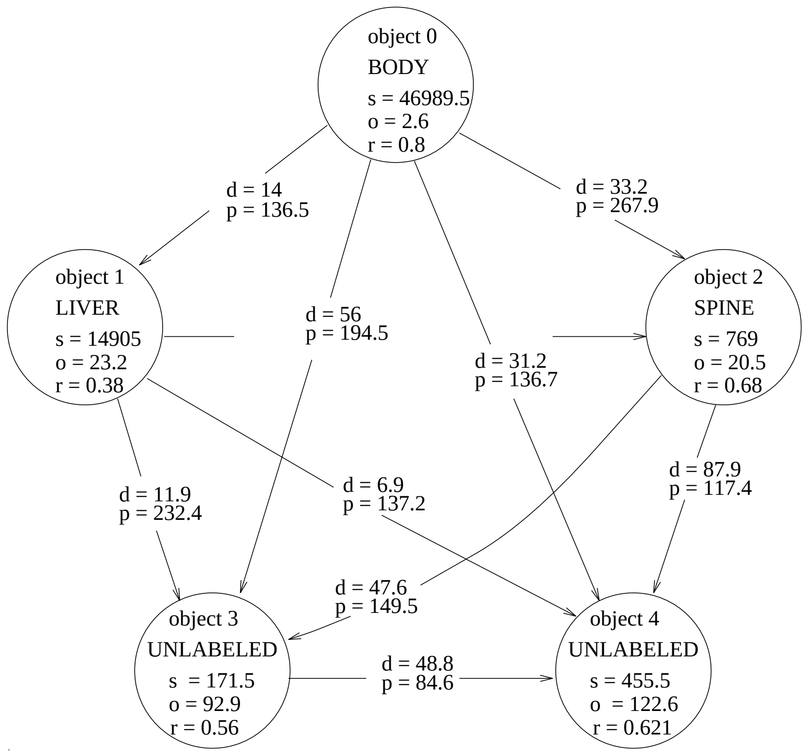

Figure 6: *Attributed graph representing the content of the example image of Figure 5.*

a pre-specified order (i.e., object 1 and its relationships with the remaining objects are taken first followed by the attribute values of object 2 etc.). For example, using the features described before, a $(k,1)$ sub-image ($k = 3$) is represented by a 24-dimensional vector: 4 objects with 3 attributes each, plus $\binom{4}{2} = 6$ relationships, with 2 attributes each. The vectors of all (k,1) sub-images are then stored in an R-tree data structure. For each vector, an image identifier corresponding to the image from which it has been derived is also stored.

Figure 7 demonstrates the proposed file structure of the data on the disk. Specifically, the IDB consists of the following parts:

- The "image file store". This is a separate disk store holding the original image files. For faster display we have also kept the segmented polygonal forms of all images (not shown in Figure 7).
- The "graph file". This is a file holding the ARGs. Each record in this file consists of an identifier (e.g., the image file name) corresponding to the image from which the ARG has been derived, and the features of each object together with its relationships with the remaining objects.

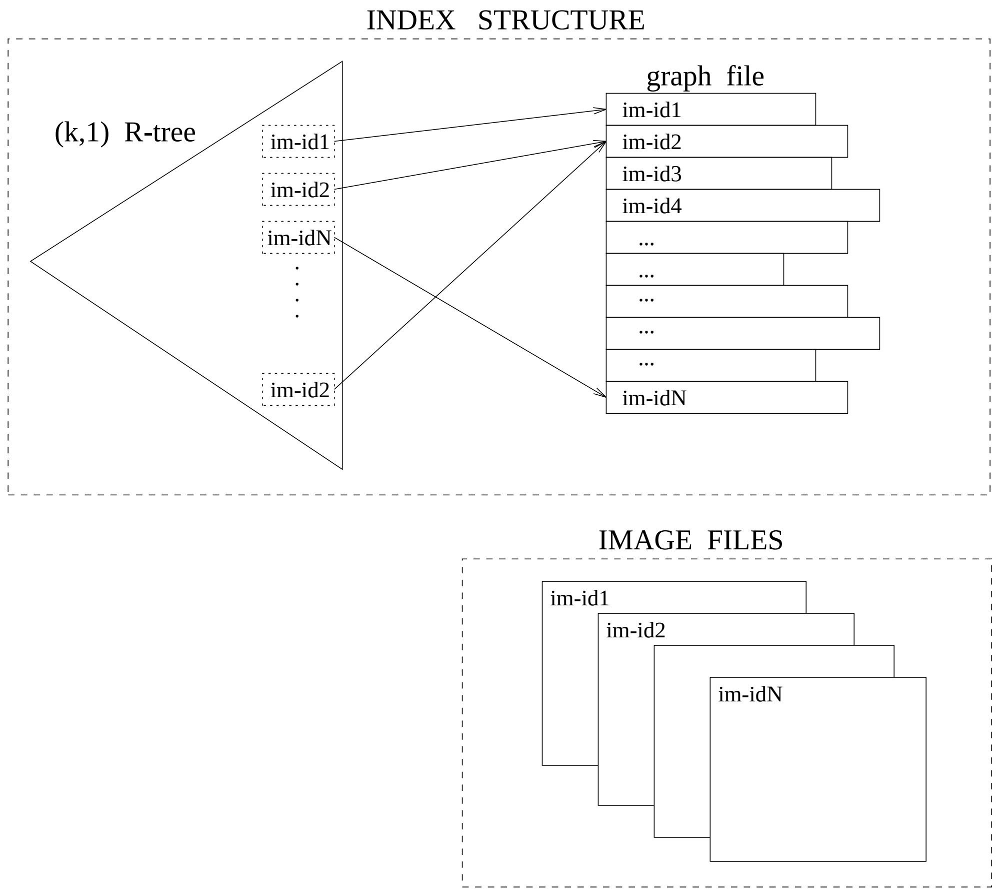

Figure 7: *Proposed file structure.*

- The R-tree data structure holding the vectors computed to all the $(k, 1)$ sub-images. We call this R-tree "$(k, 1)$ R-tree".

The graph file together with the $(k, 1)$ R-tree form the "index structure". In the IDB literature [11], the image file store and the index structure are called "physical" and "logical" database respectively. There is a plethora of alternative designs (e.g., R-trees holding vectors for $(k, 2)$ sub-images). We considered several of them and experimented with them all. The (k,1) R-tree results in the best search times for a small space overhead and is the one that we mainly focus on next. In Section 6.1 we present alternative designs along with implementation details and experiments.

4.3 Query Processing

Given a (k, u) query image and a tolerance t, we want to retrieve all images that contain a (k, u) sub-image which matches the query within tolerance t. As we show soon, the (k,1) R-tree index does not have false dismissals (i.e., all qualifying images are retrieved). However, it may return false alarms (i.e., not qualifying images) in which case, it may return a superset of the required images. A post-processing step is required to clean-up the false alarms. The generic search algorithm is as follows:

R-tree search: Issue (one or more) range queries on the (k,1) R-tree, to obtain a list of promising images (image identifiers).

Clean-up: For each of the above obtained images, retrieve its corresponding ARG from the graph file and compute the actual distance between this ARG and the ARG of the query. If the distance is less than the threshold t, the image is included in the response set.

As mentioned earlier, we decided to use the Euclidean distance (p=2 in Equation 3). However, the proposed indexing method can handle any L_p metric. For the Euclidean distance, the query region is a (hyper-)sphere; for the city-block distance (L_1) it is a diamond; for the L_∞ it is a square etc. All of the above can be handled by the R-tree (i.e., it replaces the query region by its minimum bounding rectangle and it fetches the points that fall within tolerance t).

Next, we distinguish between queries specifying one unlabeled object and queries specifying two or more unlabeled objects.

4.3.1 One Unlabeled Object

A (k,1) query specifies all labeled objects and one unlabeled. Such a query is mapped to a point in a multidimensional space of 24 dimensions (f = 24) and treated as a range query: using the Euclidean distance, we want the points in f-dimensional space that fall within a (hyper-)sphere of radius t, where t is the tolerance. The R-tree is searched and all vectors within radius t (i.e., those satisfying Equation 4) are retrieved. Feature vectors falling outside the query sphere are excluded from the answer set. Range queries on an R-tree yield neither false alarms nor false dismissals. Therefore, in this specific case of (k,1) queries there is no need for "clean-up" and the graph file need not be examined.

4.3.2 Two or More Unlabeled Objects

Here consider the case of a (k,2) query specifying 2 unlabeled objects. We break the query into two (k,1) query sub-images called "sub-queries". Then, we can apply either of the following strategies:

With-Intersection: Apply both (k,1) sub-queries to the (k,1) R-tree with tolerance t. Intersect their resulting response sets to obtain the set of common image identifiers. The ARGs corresponding to these image identifiers are retrieved from the graph file and matched with the original query to discard the false alarms.

No-Intersection: Search the (k,1) R-tree for the first sub-query with tolerance t. The second sub-query is ignored. Retrieve the resulting ARGs from the graph file and match them with the original query to discard all possible false alarms.

Both strategies introduce false alarms. The first strategy, attempts to minimize the false alarms but, involves excessive R-tree search and set intersections. The second strategy, avoids R-tree search as much as possible but, employees an expensive clean-up stage.

We can prove that the above strategies will have no false dismissals:

Lemma 1 *Replacing a (k,2) query of tolerance t with two (k,1) queries of tolerance t each and intersecting their results will give no false dismissals.*

Proof: Let Q be a (k,2) query. Its corresponding vector is $(q_1, q_2, \ldots q_f)$. Let S be a qualifying image. The vector computed to the subset of its contained objects which are similar to query objects is $(s_1, s_2, \ldots s_f)$. This vector corresponds to the best mapping $F()$ according to Equation 2. Each of the above two vectors consists of terms corresponding (a) to the labeled objects, which are denoted by l subscripts (b) to unlabeled objects, which are denoted by x subscripts for the first unlabeled object and z subscripts for the second unlabeled object, (c) to the relationships between labeled objects, which are denoted by ll subscripts, (d) to the relationships between labeled and unlabeled objects, which are denoted by lx and lz subscripts and (e) to the relationships between unlabeled objects, which are denoted by xz subscripts. Then we have:

$$\begin{array}{lclcl}
(q_{l_1} - s_{l_1})^2 & + & (q_{l_2} - s_{l_2})^2 & + \ldots & \\
(q_{x_1} - s_{x_1})^2 & + & (q_{x_2} - s_{x_2})^2 & + \ldots & \\
(q_{z_1} - s_{z_1})^2 & + & (q_{z_2} - s_{z_2})^2 & + \ldots & \\
(q_{ll_1} - s_{ll_1})^2 & + & (q_{ll_2} - s_{ll_2})^2 & + \ldots & \\
(q_{lx_1} - s_{lx_1})^2 & + & (q_{lx_2} - s_{lx_2})^2 & + \ldots & \\
(q_{lz_1} - s_{lz_1})^2 & + & (q_{lz_2} - s_{lz_2})^2 & + \ldots & \\
(q_{xz_1} - s_{xz_1})^2 & + & (q_{xz_1} - s_{xz_2})^2 & + \ldots & \leq t^2.
\end{array} \tag{5}$$

We want to prove that the above image will be retrieved by the proposed strategy. That is, we want to prove that the two (k,1) sub-queries will *each* retrieve image S. Specifically, we want to prove the following inequalities:

$$\begin{array}{lclcl}
(q_{l_1} - s_{l_1})^2 & + & (q_{l_2} - s_{l_2})^2 & + \ldots & \\
(q_{x_1} - s_{x_1})^2 & + & (q_{x_2} - s_{x_2})^2 & + \ldots & \\
(q_{ll_1} - s_{ll_1})^2 & + & (q_{ll_2} - s_{ll_2})^2 & + \ldots & \\
(q_{lx_1} - s_{lx_1})^2 & + & (q_{lx_2} - s_{lx_2})^2 & + \ldots & \leq t^2,
\end{array} \tag{6}$$

and

$$\begin{array}{lclcl}
(q_{l_1} - s_{l_1})^2 & + & (q_{l_2} - s_{l_2})^2 & + \ldots & \\
(q_{z_1} - s_{z_1})^2 & + & (q_{z_2} - s_{z_2})^2 & + \ldots & \\
(q_{ll_1} - s_{ll_1})^2 & + & (q_{ll_2} - s_{ll_2})^2 & + \ldots & \\
(q_{lz_1} - s_{lz_1})^2 & + & (q_{lz_2} - s_{lz_2})^2 & + \ldots & \leq t^2.
\end{array} \tag{7}$$

The above inequalities will definitely hold, since they can be produced from Inequality 5 by omitting some positive terms. This completes the proof of the lemma, for both strategies. □

The search algorithms for queries with more than 2 unlabeled objects are straightforward extensions of the above ideas.

4.3.3 Other Queries

The definition of similarity given in Section 2.1 serves as the basis for more complicated situations. Our proposed scheme can handle cases where the query specifies only a few of the labeled objects (e.g., we don't care for some of them). In these cases, the summation of Equation 3 excludes the features of the unspecified objects (partial match queries). The R-tree index can still handle these queries: The range of values along the unspecified axes stretches from $-\infty$ to $+\infty$.

The proposed method can also handle the case where the user considers some of the properties more important than others. We can give higher weights to these properties. If weights are used with Equation 3, the query specifies an ellipse in the feature space, instead of a sphere. The weights could even be adjusted on-the-fly by the user. Since a weighted Euclidean distance presents no additional indexing problems, we do not consider weights for the rest of this paper.

5 Experiments

To test the efficiency of our methodology, we implemented the system in C, under UNIX. As a testbed, we used a database consisting of 13500 synthetic segmented images (synthetic workload). Originally we used 20 MRI images. We segmented these images by manual tracing the contours of labeled objects. To produce a synthetic image from a given original, we allowed the objects in the original to rotate, scale and translate by a certain amount computed by a random number generator (e.g., each object is allowed to rotate between 0 and 20 degrees). Moreover, the contour points of each object were allowed to extend along the line connecting the center of mass of the object with each point. A number of unlabeled objects (at most 5 per image) having random sizes, shapes and positions was then added to the above derived images. Objects were not allowed to intersect with each other. Among the 13500 images we produced, there are 4500 images with 8 objects, 3600 with 7 objects, 2700 with 6, 1800 with 5 and 900 with 4 objects. All images contain 3 labeled objects.

We carried out several groups of experiments based on a (k,1) R-tree (i.e., holding vectors consisting of all k labeled objects and one unlabeled). Queries specifying all labeled objects and one unlabeled ((k,1) queries) are the basic queries and are used to process more complex queries specifying more unlabeled objects. The experiments were designed to:

- Study the performance of the processing of (k,1) and (k,2) queries.
- Illustrate the superiority of the proposed method over sequential scan searching.
- Study the scale-up of the proposed method.

To obtain average performance measures, the average performance of 45 characteristic queries is computed. The times reported correspond to the elapsed (i.e., "wall-clock") time by adding the "system" time and the "user" time (cpu time) as reported by the `times` system call of UNIX. The system is implemented on a SUN/470 with 32 Mbytes of main memory and a MICROPOLIS disk with average seek time less than 12msecs. When we run the experiments, no user were logged on. In all the experiments, we used 4K bytes as the page size for the R-tree. The space required for the implementation of the (k,1) R-tree is 22.5 Mbytes while, the graph file requires 5.1M bytes for storage.

5.1 Queries With One Unlabeled Object

The goal of this set of experiments is to illustrate the performance gains of our method with respect to sequential scanning of the database for the basic (k,1) queries (i.e., queries specifying all 3 labeled objects and 1 unlabeled).

Figure 8 plots the response time (averaged over 45 queries) for (k,1) queries. Our method achieves large savings, even for high values of the tolerance (for t=530 the retrieved set is almost 1000 images; presumably a user would like to retrieve 20-50 images, in which case, t must be less than 200). Even for such large queries, the response time of the proposed method is below 2 seconds. Notice that sequential scanning is always above 2 seconds.

Sequential scanning is performed by matching the vector computed to a given query with the vectors produced from each stored ARG having the same size with it. For (k,1) queries, these are at most 5 (i.e., all images contain between 1 and 5 unlabeled objects). For sequential scanning we performed the following optimization: For every image, the partial sum of the distance function of Equation 3 is continuously compared with the threshold, to achieve early rejections. Thus, as the tolerance increases, more and more images delay to be rejected, thus increasing the response time. We call this method "optimized sequential" as opposed to the "naive sequential" method (i.e., the sum of Equation 3 is computed first and then compared with the tolerance). For the naive sequential scan the response time was almost constant at 13.5 secs. Henceforth, the optimized sequential scan is used in the experiments.

The shape of the curves can be justified as follows: For the optimized sequential scanning, the search time increases with the tolerance because of the optimization we described earlier. For

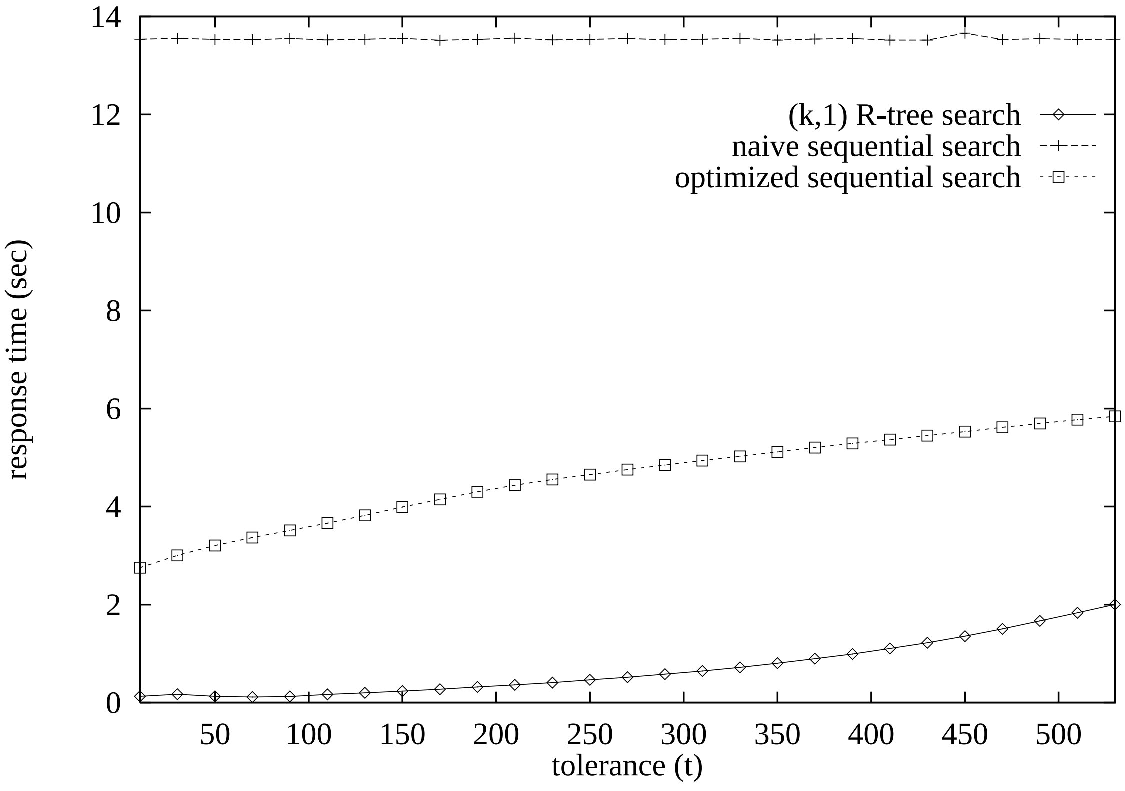

Figure 8: *Average retrieval response time as a function of the tolerance (t) for (k,1) queries corresponding to (a) search on a (k,1) R-tree, (b) naive sequential scan and (c) optimized sequential scan of the database.*

the proposed method, the shape of the curve resembles an exponential curve. This is expected, because the number of qualifying images increases exponentially with the tolerance.

Figure 9 plots the response time as a function of the number of qualifying images (response-set size). The labels indicate the corresponding value of the tolerance. This figure contains almost the same information with Figure 8, showing in addition the response-set size.

5.2 Queries With Two Unlabeled Objects

The next set of experiments examines (k,2) queries (i.e., queries specifying all 3 labeled objects and 2 unlabeled). Figure 10 plots the average retrieval response time of (k,2) queries as as a function of the tolerance t.

A (k,2) query breaks into two (k,1) sub-queries. As explained in Section 4.3, we have two

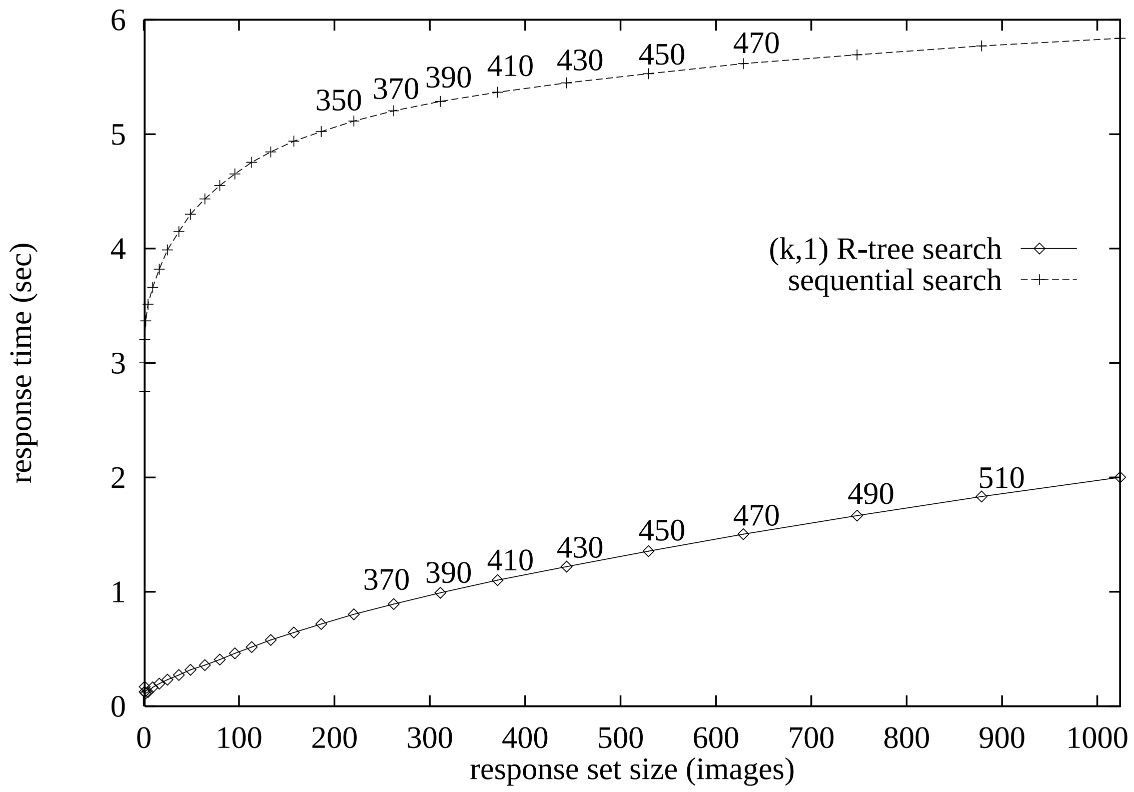

Figure 9: *Average retrieval response time as a function of the average retrieval response set size for (k,1) queries. The labels denote the corresponding values of the tolerance (t).*

alternative methods to search the (k,1) R-tree:

With-Intersection: Both (k,1) sub-queries are processed by the R-tree.

No-Intersection: Only the first (k,1) sub-query is processed by the R-tree.

The experimental results show that above two methods are roughly comparable with respect to response time. The recommended method is the second one, since it is faster than the first one. The shape of the curves is also expected, resembling curves that are exponential as a function of the tolerance. The sequential scanning was the slowest, even for large values of the tolerance, ranging from 6 seconds to 15 seconds. Its response time is not constant, because of the optimization we described in the case of (k,1) queries. The rest of the experiments in this section concentrate on (k,1) queries.

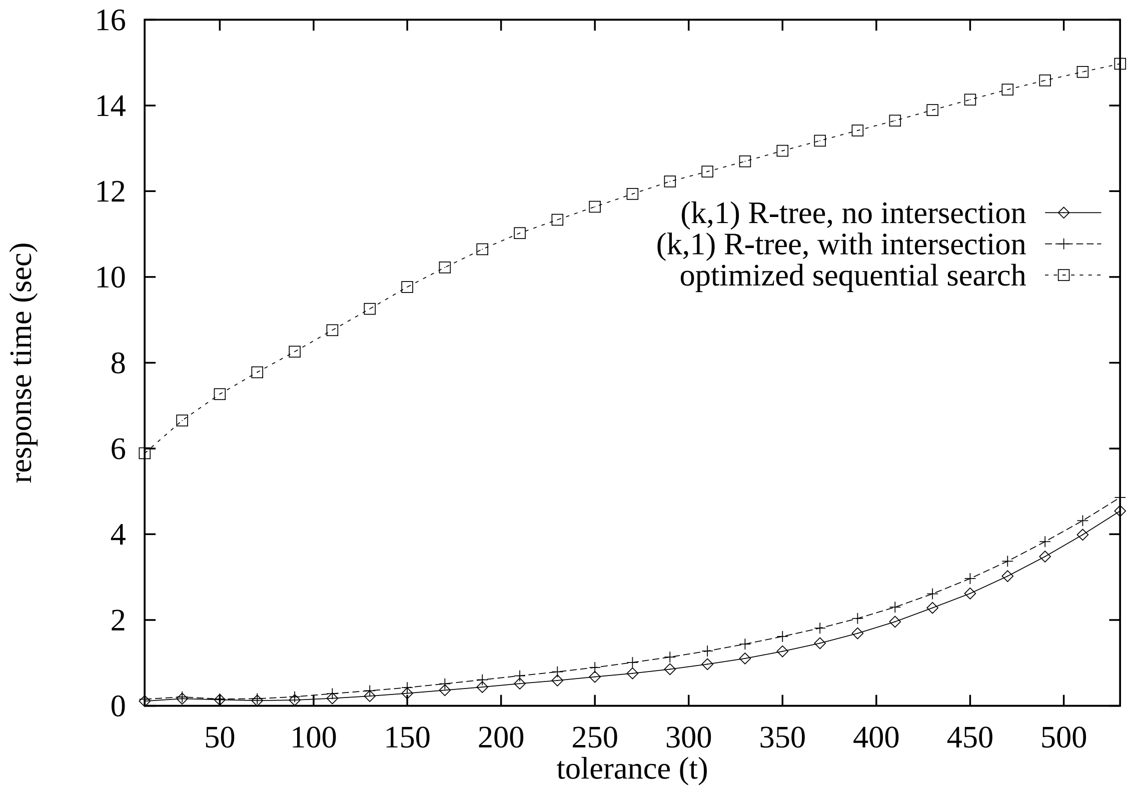

Figure 10: *Average retrieval response time as a function of the tolerance (t) for (k,2) queries corresponding to (a) search on a (k,1) R-tree utilizing one (k,1) sub-query, (b) search on a (k,1) R-tree utilizing both (k,1) sub-queries and (c) optimized sequential scan of the database.*

5.3 Scale-Up

In this set of experiments we examined the performance of the proposed method as a function of the database size N (number of stored images). Figure 11 plots the average retrieval response time for the sequential scan and for the proposed method. The tolerance is set to $t = 310$. Notice that the performance gap between the two methods widens as the database grows. Therefore, the proposed method becomes increasingly attractive for larger databases.

5.4 Examples of Retrievals

Figure 12 demonstrates a characteristic example of a (k,1) query (left) and of a retrieved image (right). The cost of matching their corresponding 24-dimensional vectors is 219.4. Observe that, both labeled and unlabeled objects in the query and the retrieved image respectively, have approx-

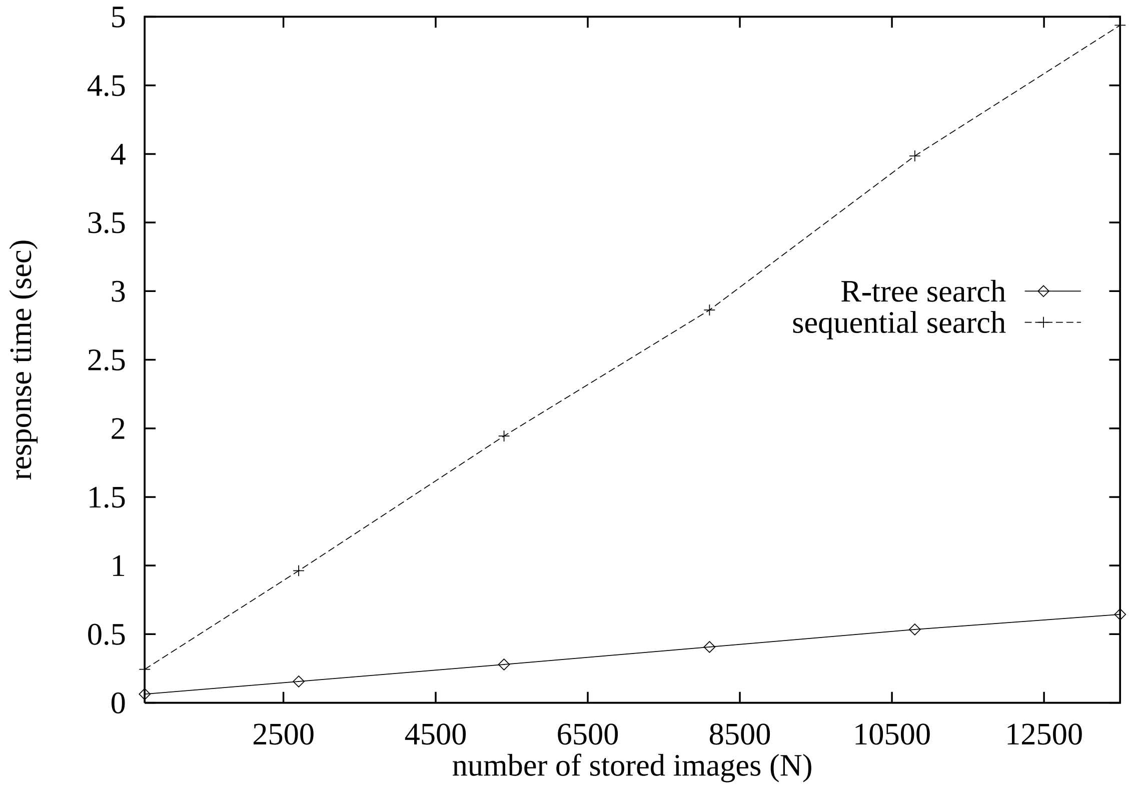

Figure 11: *Average retrieval response time corresponding to a fixed value of the tolerance ($t = 310$) as a function of the number of images (N) stored in the database.*

imately similar characteristics and similar spatial relationships. The retrieved image contains also one additional unlabeled object (object 4) not specified by the query.

6 Discussion - Observations

There are several interesting observations, as well as a plethora of optimization techniques that can be used. In this section we present a discussion on alternative index designs, as well as some observations, which may lead to some interesting research in the future.

6.1 Alternative Index Organizations

We can use the (k,1) R-tree to answer queries with any number of unlabeled objects. However, it is natural to wonder what would be the performance of a (k,2) R-tree. This R-tree would contain

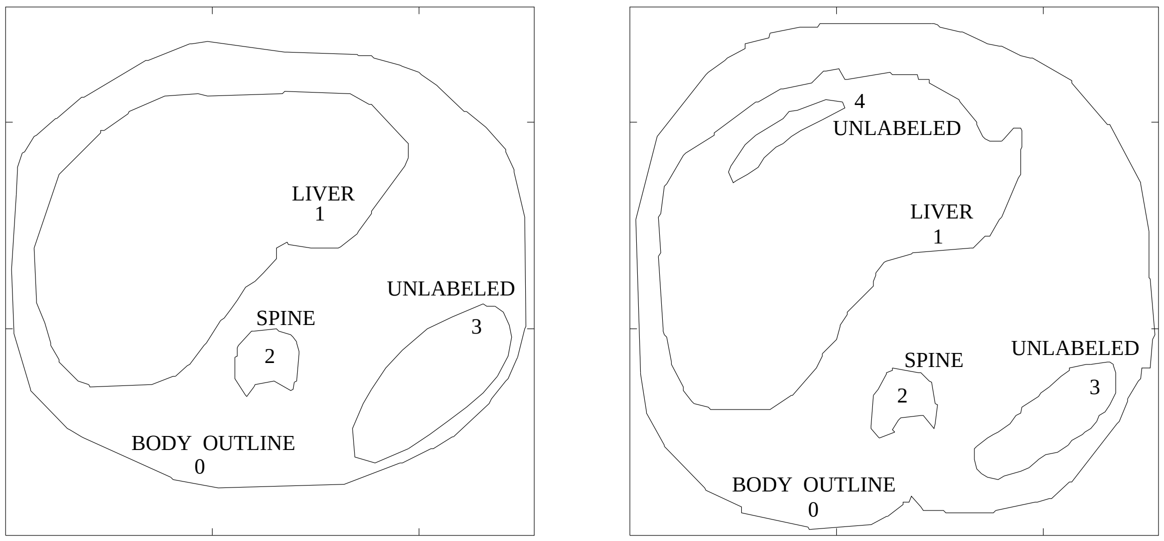

Figure 12: *Example of a query image specifying all 3 labeled and 1 unlabeled object (left) and example of a retrieved image (right).*

all the feature vectors of the (k,2) sub-images derived from all the images stored in the database. Then, an image with 5 unknown objects would yield $\binom{5}{2}$ vectors, all of which would be stored in the k-2 R-tree. Generalizing, we can maintain a (k,3) R-tree, a (k,4) R-tree and so on. Notice, however, two important points:

- A (k,2) R-tree can not answer (k,1) queries. In general, a (k, u) R-tree can not answer (k, u') queries, if $u' \leq u$. Therefore, the (k, u) R-tree has to be maintained in addition to the (k,1) R-tree, the (k,2) R-tree, ... the $(k, u - 1)$ R-tree.

- The space overhead of a (k, u) R-tree increases fast with u: For example, an image with 6 unlabeled objects, will yield $\binom{6}{1} = 6$ entries for the (k,1) R-tree, $\binom{6}{2} = 15$ and $\binom{6}{3} = 20$ entries for the (k,2) and the (k,3) R-tree respectively.

We implemented the (k,2) R-tree and measured its search time for (k,2) queries. Figure 13 shows the response time of the (k,2) R-tree ($f = 35$), as a function of the tolerance t for the same setting as in Section 5 with the rest of the experiments. Its response time is very good, mainly because it does not require clean-up. However, the required space overhead is large: 52.2 Mbytes as opposed to 22.5 Mbytes of the (k,1) R-tree. Given the large space requirements and its difficulty in answering (k,1) queries, the (k,2) R-tree is not recommended. For the same reasons, other R-trees with more unlabeled objects are also not recommended.

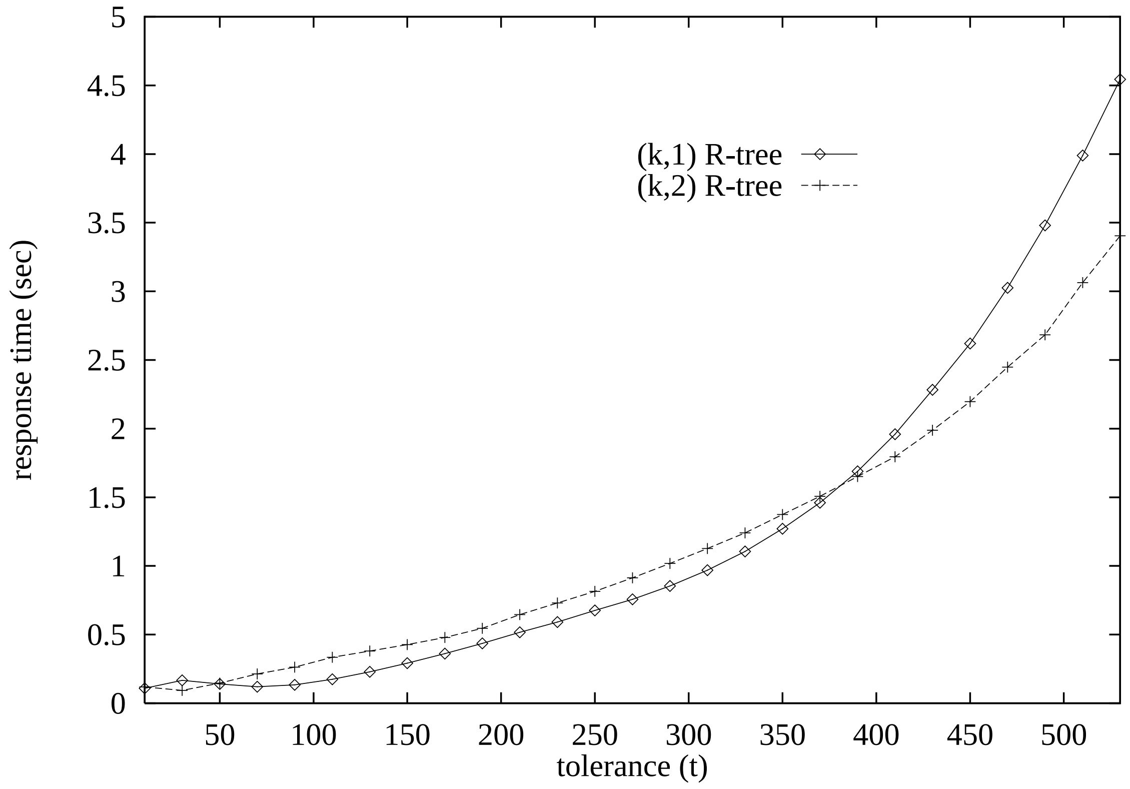

Figure 13: *Average retrieval response time as a function of the tolerance (t) for (k,2) queries corresponding to search on (a) a (k,1) R-tree and (b) a (k,2) R-tree.*

6.2 Using Fewer Features for Indexing

Our proposed method has no false dismissals that is, it will retrieve each and every hit that sequential scanning would have retrieved, only faster. For the $(k,1)$ queries, this is obvious, because our R-tree index contains all the attributes that the distance function needs. However, we can show that we can omit some of the attributes, and still guarantee no false dismissals, introducing some false alarms (and subsequently, a more expensive clean-up step), to achieve a smaller, hopefully faster, R-tree.

To guarantee no false dismissals, the distance between two sub-images in a lower dimensionality space should under-estimate their distance when all attributes are used. This is always true, since the omitted attributes correspond to terms omitted from Equation 3; these terms are all positive and, therefore, the distance between the two images is under-estimated. For both cases the tolerance is assumed to be t. Then, every qualifying sub-image will be retrieved with both methods.

It is beneficial to keep "useful" attributes (e.g., the most discriminatory attributes). Conversely, dropping "useless" attributes will introduce a few additional false alarms. A useless feature would be one having the same value in every image. For example, we can ignore the attributes of the labeled objects "spine" and "body outline" from the ARG of Figure 6, since these objects look similar in most images. This will introduce false alarms, but no false dismissals. Notice that, an R-tree with some attributes omitted is still able to answer (k,2) queries (as well as (k,3), (k,4) etc. queries) without false dismissals.

Compared to the original method, the trade-off is as follows: If we keep fewer attributes in the (k,1) R-tree, more vectors will fit per R-tree node resulting in higher fan-out and, therefore, into a smaller, shallower and faster R-tree. The penalty we have to pay is the increased number of false alarms that the post-processing step will have to deal with.

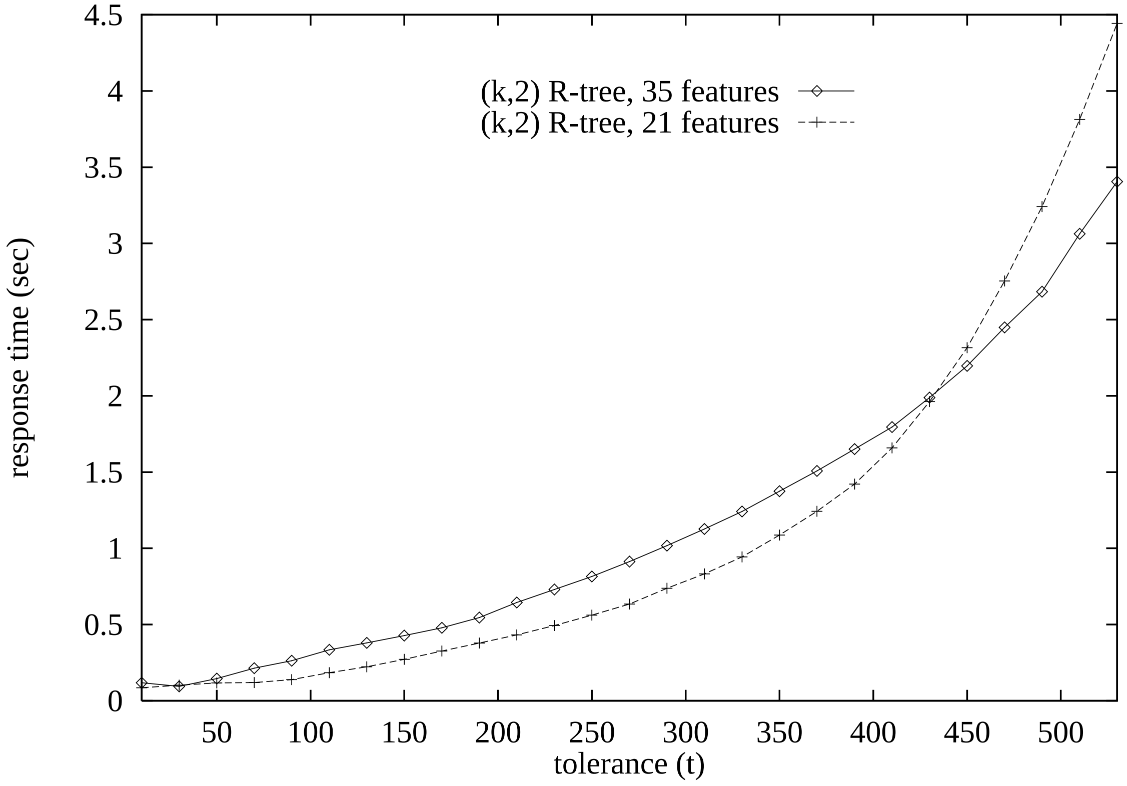

Figure 14: *Average retrieval response time as a function of the tolerance (t) for (k,2) queries corresponding to search on a (k,2) R-tree utilizing (a) all 35 attributes and (b) 21 attributes.*

In order to fully exploit this method, it is necessary to develop a way to distinguish the useful attributes from the useless ones. An in-depth analysis of such methods is outside the scope of this paper. Here we only outline two possible solutions:

- A domain expert may be used to pinpoint the useful attributes.
- In case of a static (or slowly changing) database, we can use the Karhunen-Loeve transform, which, given a static set of f attributes, creates f new attributes sorted in "usefulness" order. Keeping the first few attributes we can probably retain most of the information needed.

We ran some experiments to determine whether the lower dimensionality R-trees show any promise. By omitting attributes of labeled objects, we reduced the dimensionality of the (k,1) R-tree from 24 to 12, and of the (k,2) R-tree from 35 to 21. The original (k,1) R-tree with $f = 24$ was faster than the R-tree with $f = 12$ for (k,1) queries. However, as observed from Figure 14, this is not always the case for (k,2) queries. Therefore, we believe that, if the most useful attributes are carefully selected, the resulting R-tree will probably be faster in cases of very high dimensionality spaces.

6.3 Effective/Fractal Dimension

Here we focus on (k,1) queries. As observed in section 5.1, the response time increases exponentially with the tolerance t. Figure 15 shows the number of qualifying (k,1) sub-images as a function of the tolerance, in doubly logarithmic scales. It also plots the line of a linear regression. It is interesting to note that the line gives a good approximation. The slope of the line is $\approx$3.

The slope of such diagrams is related to the "correlation fractal dimension" [52], which is similar to the "effective dimensionality" [24]. This implies that our data points in feature space follow a skewed distribution: If they were uniformly distributed in a f-dimensional feature space, then the slope should be f (i.e., the number of neighbors within distance t should increase with the t^f). The fact that it increases like t^3 means that the point-set behaves like a 3-dimensional object embedded in a f-dimensional space. This observation can be used in the following two ways:

(a) Dimensionality reduction techniques, such as the Karhunen-Loeve (K-L) transform, can be used to reduce the dimensionality of the R-tree from $f = 24$ to 3 (i.e., by keeping the first three attributes of the (K-L) transform). In cases where a large number of image attributes are used for indexing, dimensionality reduction might make the R-tree faster.

(b) It may help us predict the selectivity (i.e., response-set size) and subsequently, the response time of our method. In [21] we show how to estimate the performance of R-trees using the concept of fractal dimension, which, in our case, is ≈ 3.

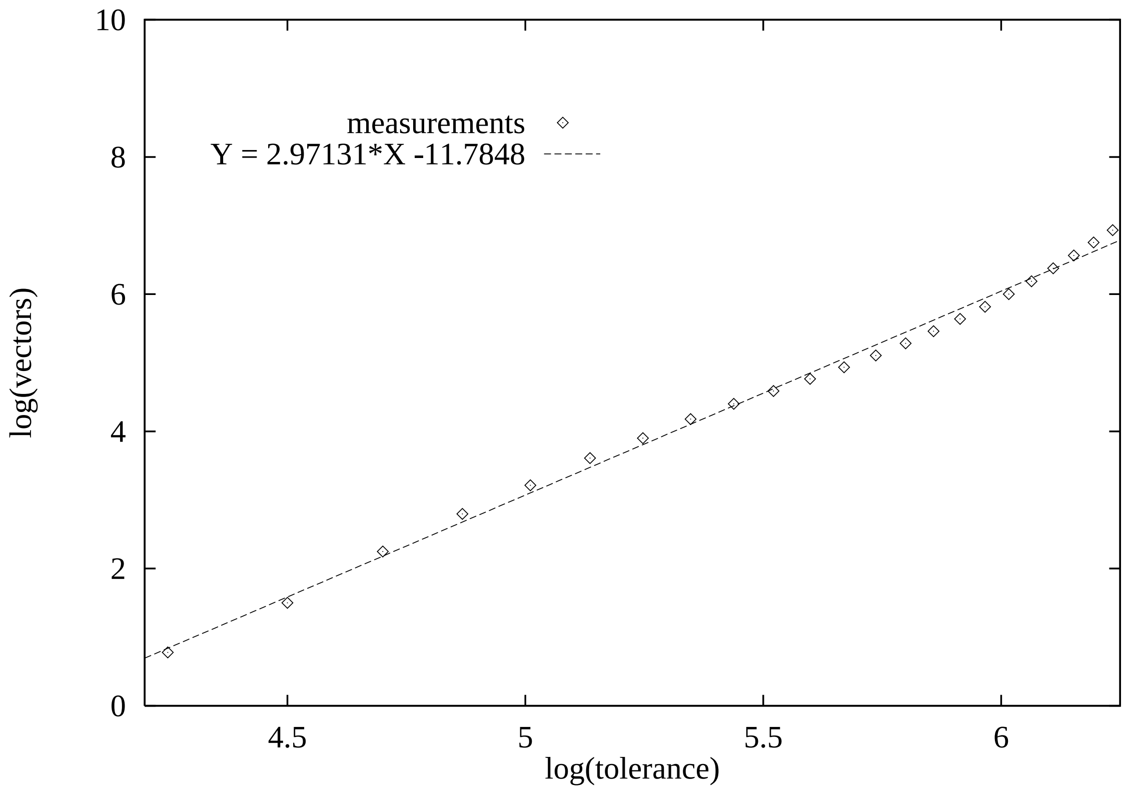

Figure 15: *Logarithm of the number of qualifying vectors as a function of the logarithm of the tolerance t for (k,1) queries.*

Skewed distributions of points in feature space should be common to almost any set of images, since image content representations usually include correlated attributes. Thus, we expect that the "effective/fractal" dimensionality will be small in other IDB applications too. As an example of correlated attributes, consider the case of an object that contains other objects. The former is always bigger than the objects it contains. In turn, its contained objects are usually close to each other. This is the case with the medical CT and MRI scans we used. In all these cases, we believe that the estimation of the fractal/effective dimension will give useful information to the system builders, both to design the index structure, as well as for query optimization.

7 Conclusions

In this paper, we proposed a method to handle approximate searching by image content in large image databases. Our approach allows for continuous, quantitative estimates of similarity. Older methods, such as 2-D strings [15], give binary (i.e., "yes/no") answers while, others are time consuming and cannot be used to support retrievals in large databases [37]. In addition, image content representation methods based on strings have been proven to be ineffective in capturing image content and may yield inaccurate retrievals. Attributed relational graphs (ARGs) provide an effective means for image content representation. However, retrievals based on attributed relational graphs are inefficient. This is mostly due to the complexity of search [54]. In addition, search is exhaustive. In this work, we proposed a method for the indexing of stored attributed relational graphs. We make the assumption that certain labeled objects can be identified in all images. This situation is common to images found in many application domains including medicine, remote sensing, microscopy, robotics etc. In this work, we focused our attention on medical images (i.e., tomographic scans of the body).

Our method allows similarity search to be performed on both labeled and unlabeled (i.e., not identified) objects. Indexing is performed by decomposing each input image into sets of objects, called "sub-images", containing all labeled objects and a fixed number of unlabeled. All sub-images are mapped to points in a multidimensional feature space implemented as an R-tree. Image database search is then transformed into spatial search. We provide experimental results on a synthetic, but realistic database. The experimental results are a good support to the claims of efficiency. We show that the proposed method outperforms sequential scanning significantly (i.e., up to an order of magnitude), never missing a hit (no false dismissals are allowed).

Future work includes (a) the examination of dimensionality-reduction methods, (b) the design of a powerful query language supporting the processing of various types of image queries, (c) the extension of this method to work on a parallel machine supporting parallel disc access and (d) the development of an object oriented data model and system designed according to the principles of the system of [46], capable of handling various types of medical images and taking advantage of the time efficiency of the methodology proposed in this paper.

References

[1] Walid G. Aref and Hanan Samet. Optimization strategies for spatial query processing. *Proc. of VLDB (Very Large Data Bases)*, pages 81–90, September 1991.

[2] Jeffrey R. Bach, Santanu Paul, and Ramesh Jain. A Visual Information Management System for the Interactive Retrieval of Faces. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):619–627, August 1993.

[3] S. Back, H. Neumann, and H. S. Stiehl. On Segmenting Computed Tomograms. In *Computer Assisted Radiology, CAR89*, pages 691–696, Berlin, June 1989.

[4] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice Hall, 1982.

[5] Nobert Beckmann, Hans-Peter Kriegel, Ralf Scneider, and Bernhard Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD*, pages 322–331, Atlantic City, NJ, May 1990.

[6] J.L. Bentley. Multidimensional binary search trees used for associative searching. *CACM*, 18(9):509–517, September 1975.

[7] Jon Louis Bentley and Jerome H. Friedman. Data Structures for Range Searching. *ACM Computing Surveys*, 11(4):397–409, December 1979.

[8] Elizabeth Binaghi, Isabella Gagliardi, and Raimondo Schettini. Indexing and Fuzzy Logic-Based Retrieval of Color Images. In *Visual Database Systems, II, IFIP Transactions A-7*, pages 79–92. Elsevier Science Publishers, 1992.

[9] Robert C. Bolles and Ronald A. Cain. Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method. *The International Journal of Robotics Research*, 1(3):57–82, 1982.

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

[11] Shi-Kuo Chang. *Principles of Pictorial Information Systems Design.* Prentice Hall International Editions, 1989.

[12] Shi-Kuo Chang and King-Sun Fu. A Relational Database System for Images. In Shi-Kuo Chang and King-Sun Fu, editors, *Pictorial Information Systems*, pages 288–321. Springer-Verlag, 1980.

[13] Shi-Kuo Chang and Arding Hsu. Image Information Systems: Where Do We Go From Where? *IEEE Transactions on Knowledge and Data Engineering*, 4(5):431–442, 1992.

[14] Shi-Kuo Chang, Erland Jungert, and Y. Li. Representation and Retrieval of Symbolic Pictures Using Generalized 2-D Strings. In *SPIE Proceedings, Visual Communications and Image Processing*, pages 1360–1372, Philadelphia, November 1989.

[15] 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.

[16] S. Christodoulakis, M. Theodoridou, F. Ho, M. Papa, and A. Pathria. Multimedia Document Presentation, Information Extraction and Document Formation in MINOS: A Model and a System. *ACM Transactions on Office Information Systems*, 4(4):345–383, October 1986.

[17] Silvana Dellepiane, Giovanni Venturi, and Gianni Vernazza. Model Generation and Model Matching of Real Images by a Fuzzy Approach. *Pattern Recognition*, 25(2):115–137, 1992.

[18] M. A. Eshera and King-Sun Fu. A Graph Distance Measure for Image Analysis. *IEEE Transactions on Systems Man and Cybernetics*, SMC-14(3):353–363, 1984.

[19] M. A. Eshera and King-Sun Fu. An Image Understanding System Using Attributed Symbolic Representation and Inexact Graph-Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):604–618, 1986.

[20] Christos Faloutsos, Ron Barber, Myron Flickner, Wayne Niblack, Dragutin Petkovic, and William Equitz. Efficient and effective querying by image content. *J. of Intelligent Information Systems*, 3(3/4):231–262, July 1994.

[21] Christos Faloutsos and Ibrahim Kamel. Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. *Proc. ACM SIGACT-SIGMOD-SIGART PODS*, pages 4–13, May 1994. Also available as CS-TR-3198, UMIACS-TR-93-130.

[22] 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.

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

[24] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990. 2nd Edition.

[25] I. Gargantini. An effective way to represent quadtrees. *Comm. of ACM (CACM)*, 25(12):905–910, December 1982.

[26] O. Gunther. The cell tree: an index for geometric data. Memorandum No. UCB/ERL M86/89, Univ. of California, Berkeley, December 1986.

[27] Antonin Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of ACM SIGMOD*, pages 47–57, June 1984.

[28] K. Hinrichs and J. Nievergelt. The Grid-File: A Data Structure to Support Proximity Queries on Spatial Objects. Technical Report 54, Institut fur Informatik, ETH, Zurich, July 1983.

[29] Hou, Hsu, Liu, and Chiu. A content-based indexing technique using relative geometry features. *SPIE 92*, 1662:59–68, 1992.

[30] H. V. Jagadish. Linear Clustering of Objects with Multiple Attributes. In *Proceedings of ACM SIGMOD*, pages 332–342, Atlantic City, May 1990.

[31] H. V. Jagadish. A retrieval technique for similar shapes. In *International Conference on Management of Data, SIGMOD 91*, pages 208–217, Denver, CO, May 1991. ACM.

[32] R. Jain and W. Niblack. Nsf workshop on visual information management, February 1992.

[33] Ioannis Kapouleas. Segmentation and Feature Extraction for Magnetic Resonance Brain Image Analysis. In *Proceedings of 10th International Conference on Pattern Recognition*, pages 583–590, Atlantic City, New Jersey, June 1990.

[34] Toshikazu Kato, Takio Kurita, Nobuyuki Otsu, and Kyoji Hirata. A Sketch Retrieval Method for Full Color Image Database - Query by Visual Example. In *Proceedings of 11th International Conference On Pattern Recognition*, pages 530–533, The Hague, The Netherlands, August 1992.

[35] 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.

[36] 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.

[37] Suh-Yin Lee and Fang-Jung Hsu. Spatial Reasoning and Similarity Retrieval of Images using 2D C-Sstring Knowledge Representation. *Pattern Recognition*, 25(3):305–318, 1992.

[38] Suh-Yin Lee, Man-Kwan Shan, and Wei-Pang Yang. Similarity Retrieval of Iconic Image Databases. *Pattern Recognition*, 22(6):675–682, 1989.

[39] David B. Lomet and Betty Salzberg. The hb-tree: a multiattribute indexing method with good guaranteed performance. *ACM TODS*, 15(4):625–658, December 1990.

[40] Ramakant Nevatia and Keith E. Price. Locating Structures in Aerial Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(5):476–484, September 1982.

[41] Jack A. Orestein. Spatial Query Procesing in an Object Oriented Database System. In *ACM Proceedings SIGMOD 86*, pages 326–336, Washington, May 1986.

[42] S. C. Orphanoudakis, C. Chronaki, and S. Kostomanolakis. I2C: A System for the Indexing, Storage and Retrieval of Medical Images by Content. Technical Report 113, Institute of Computer Science, Foundation for Research and Technology - Hellas, Heraklion, Greece, January 1994. To appear in the Journal of Medical Informatics.

[43] Stelios C. Orphanoudakis, Euripides G. Petrakis, and Petros Kofakis. A Medical Image DataBase System for Tomographic Images. In *Proceedings of Computer Assisted Radiology, CAR89*, pages 618–622, Berlin, June 1989.

[44] Michael Otterman. Approximate matching with high dimensionality r-trees. M.Sc. scholarly paper, Dept. of Computer Science, Univ. of Maryland, College Park, MD, 1992. supervised by C. Faloutsos.

[45] Euripides G.M. Petrakis and Stelios C. Orphanoudakis. A Generalized Approach for Image Indexing and Retrieval Based Upon 2-D Strings. In Shi-Kuo Chang, Erland Jungert, and Genoveffa Tortora, editors, *Visual Reasoning*. Plenum Publishing Co., 1993. To be publised. Also available as FORTH-ICS/TR-103.

[46] Euripides G.M. Petrakis and Stelios C. Orphanoudakis. Methodology for the Representation, Indexing and Retrieval of Images by Content. *Image and Vision Computing*, 11(8):504–521, October 1993.

[47] F. Rabitti and P. Savino. An Information Retrieval Approach for Image Databases. In *Proceedings of the 18th International Conference on VLDB*, pages 574–584, Vancuver, British Columbia, Canada, August 1992.

[48] A.V. Raman, S. Sarkar, and K. L. Boyer. Hypothesizing Sstructures in Edge-Focused Cerebral Magnetic Images Using Graph-Theoretic Cycle Enumeratation. *CVGIP: Image Understanding*, 57(1):81–98, January 1993.

[49] John T. Robinson. The k-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes. In *Proceedings of ACM SIGMOD*, pages 10–18, 1981.

[50] N. Roussopoulos and D. Leifker. Direct spatial search on pictorial databases using packed r-trees. *Proc. ACM SIGMOD*, May 1985.

[51] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1989.

[52] Manfred Schroeder. *Fractals, Chaos, Power Laws: Minutes From an Infinite Paradise.* W.H. Freeman and Company, New York, 1991.

[53] Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The R^+-tree: A Dynamic Index for Multidimensional Objects. In *Proceedings of 13th International Confernece on VLDB*, pages 507–518, England, September 1987.

[54] 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.

[55] Michael J. Swain and Dana H. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[56] Hemant D. Tagare, Conrade C. Jafee, and James S. Duncan. Arrangement: A Spatial Relation for Describing and Comparing Part Embeddings. In *Proceedings of 11th International Conference On Pattern Recognition*, pages 91–94, The Hague, The Netherlands, August 1992.

[57] Hideyuki Tamura and Naokazu Yokoya. Image Database Systems: A Survey. *Pattern Recognition*, 17(1):29–49, 1984.