Image Indexing Based on Spatial Similarity

Euripides G.M. Petrakis[*]
Dept. of Elect. and Comp. Eng.
Technical University of Crete
Chanea, Greece
e-mail: petrakis@ced.tuc.gr

Christos Faloutsos [†]
Dept. of Comp. Science and
Inst. for Systems Research (ISR)
Univ. of Maryland, USA
e-mail: christos@cs.umd.edu

King-Ip (David) Lin
Dept. of Mathematical Sciences
University of Memphis
Tennessee, USA
e-mail: linki@msci.memphis.edu

June 21, 2002

Abstract

We introduce *ImageMap*, as a method for indexing and similarity searching in Image DataBases (IDBs). ImageMap answers "queries by example", involving any number of objects or regions and taking into account their inter-relationships. We adopt the most general image content representation, that is *Attributed Relational Graphs* (ARGs), in conjunction with the well-accepted *ARG editing distance* on ARGs. We tested ImageMap on real and realistic medical images. Our method not only provides for visualization of the dataset, clustering and data-mining, but it also achieves up to 1,000-fold speed-up in search over sequential scanning, with zero or very few false dismissals.

Index Terms: Image database, similarity retrieval, query by example, image content, attributed relational graph, editing distance, image indexing.

1 Introduction

In many application domains, images comprise the majority of acquired data. The management of large volumes of digital images has generated additional interest in methods and tools for real time archiving and retrieval of images by content. Potential applications of content based image retrieval include: (a) *Medical Applications*: Given a patient's examination (e.g., a CT or MRI image) a clinician would like to retrieve similar cases from the medical archive. Such content-based retrievals would not only yield cases of patients with similar examinations (i.e., images showing similar objects or regions with similar spatial relationships) and similar diagnosis but also, cases of patients with similar image examinations and different diagnoses [1, 2], (b) *Criminal Investigation*:

[*]Corresponding author.

[†]On leave at Carnegie Mellon. Partially supported by the National Science Foundation under grants EEC-94-02384, IRI-9625428.

A police officer would like to retrieve images resembling the face or the fingerprint image of a suspect [3, 4], (c) *Trademark/Copyright Detection*: Given a trademark image, we would like to detect if it already exists in the database [5]. Education, home entertainment systems, remote sensing, astronomy, cartography and defense, are just a few more applications with great interest. Consequently, content-based image indexing and retrieval has become the object of intensive research activities over the past few years [6, 7, 8].

To support retrievals by image content, images must be analyzed prior to storage so that, descriptions of their content can be extracted and stored in the Image DataBase (IDB) together with the original images. These descriptions are then used to search the IDB to determine which images satisfy the query selection criteria.

In this work, we deal with the following problem:

- We have a collection of N images (medical images in this work) as in Figure 4(left).
- Each image has been segmented (automatically or manually) to one or more objects or regions. For example, Figure 4(right) contains five regions (i.e., the body, the spine, the liver, a tumor and an unknown or not recognized region) which are represented by their bounding closed polygons.
- We are also given a distance function between two images (i.e., sets of regions). The more similar the objects or regions and their relationships in the two images are, the lower the value of the distance function.
- The queries are "by example": The user specifies a desirable image with one or more regions (e.g., "*find the examinations which are similar to Smith's examination*").
- The system has to return all the images below a distance threshold, or the most similar images.

Our goal is to respond to these queries *fast*. A secondary goal is to support visualization and data-mining (eg., study of the clustering properties of the set of images). A critical point in the overall process is to use a good distance function between two images.

We summarize the contributions of this work in the following:

- We propose *ImageMap*, a method that achieves up to 1,000-fold speed-up over sequential scanning with little, if any, false dismissals.
- ImageMap, maps images into low-dimensionality points allowing visualization, clustering and other data-mining operations.
- We introduce to the database community a general image representation methodology from the Machine Vision research, namely, *Attributed Relational Graphs* (ARGs) and the Eshera-Fu ARG editing distance function [9] to compute the distance between two ARGs. This is the most general among other known functions [10]. We propose a more efficient implementation of it.
- We introduce a generic indexing technique to ARGs. Existing methods assume special kinds of ARGs [11, 2], require prohibitively large space for storage [12], are static (i.e., they cannot handle insertions or deletions), do not guarantee high recall (accuracy) and assume that the stored images are similar [13]. Our proposed approach handles all these issues.

We experimented with tomographic images but our method can handle any image type, such as real world images and queries in the form *"find images showing a tree close to a house"*. Image content is given in terms of features specific to the shape of individual objects or regions and in terms of features of spatial relationships. However, our method is independent of any specific kind of features.

The rest of this paper is organized as follows: A review of related work in the areas of Computer Vision and DataBases is presented in Section 2. A short presentation of the underlying theory is given in Section 3. The proposed methodology is presented in Section 4. In Section 5, experimental results are given and discussed. The conclusions and the issues for future research are discussed in Section 6. Finally, the Eshera-Fu algorithm is discussed shortly in Appendix A. In addition, our implementation developed as part of this work, which presents a more efficient distance algorithm is presented in Appendix B.

2 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. Interesting surveys have been published in [6, 7, 8]. 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 [14], 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. The proposed method is a contribution towards this direction.

2.1 Content Based Image Retrieval

Several approaches to the problem of content-based image management have been proposed and some have been implemented on research prototypes and commercial systems.

In the Virage system [15], image content is given primarily in terms of properties of color and texture. The QBIC system of IBM [16] utilizes a retrieval method for queries on color, texture and shape. The Photobook [17], the system developed at the MIT Media Lab, supports queries by image content in conjunction with text queries. The CAFIIR system [3] proposes the "iconic index tree" to accelerate the search on facial images. A desirable feature common to many systems is the adaptive behavior to retrievals through user relevance feedback and iterative query refinement (e.g., [8]).

Additional work on IDB systems and content based image retrieval include, the work by Ratha, Karu, Chen and Jain for fingerprints [4], the work by Mehrota and Gary for shapes [18], and the work by Mehtre, Kankanhalli and Lee for trademark authentication [5].

The methods referred to above can be extended to handle video: A video is regarded as a sequence of related image frames from which the "keyframes" (i.e., frames representative of the video content) can be selected and used for content-based indexing and retrieval (e.g., [15, 16, 19]). Motion fields can also be used to relate neighboring keyframes in the retrieval process [20]. Del Bimbo, Vicario and Zingoni proposed an approach based on spatio-temporal logic [21].

Focusing mainly on color, texture and shape, the work referred to above does not show how to handle multiple objects or regions in example queries, nor their inter-relationships. 2D strings [22] and their variants [23] are examples of work focusing mainly on spatial relationships. VisualSEEk [24] combines 2D strings with texture and color properties of selected regions. 2D string matching

give binary (i.e., "yes/no") answers and may yield "false alarms" (non-qualifying images) and "false dismissals" (qualifying but not retrieved images) [25]. Gudivada and Raghavan use only relative orientation to describe spatial relationships [26]. In our previous work [2], we used R-trees for the indexing of ARGs holding any kind of features and we allowed for continues, quantitative estimates of similarity. However, we did not allow for missing or extra regions in images and we required that at least some of the regions are labeled and present in every image. In the current work, we relax these restrictions.

2.2 Spatial Access Methods (SAMs)

We can achieve faster-than-sequential searching by using "spatial access methods". These are file structures to manage a large collection of multidimensional points (or rectangles) 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.

Several spatial access methods have been proposed forming the following classes: (a) Methods that transform rectangles into points in a higher dimensionality space [27]; (b) Methods that use linear quad-trees or, equivalently, the "z-ordering" or other "space filling curves" [28]; and finally, (c) Methods based on trees, like the R-tree [29] and its derivatives (R*-tree [30] etc.). Recent extensions for high-dimensions include the X-tree [31] and the SR-trees [32].

Methods referred to as "metric trees" or "distance trees" are based on the idea of indexing using distance information [33, 34, 35]. All these methods try to exploit the triangle inequality in order to prune the search space on a range query. However, none of them tries to map images into points in a "target space" (also known as "feature space"), nor to provide a tool for visualization. Besides, most of these methods require expensive preprocessing for building a tree index structure.

An alternative to metric trees is *FastMap* [36]. FastMap is an algorithm that takes in a set of N data items (e.g., images), together with a distance function $D()$, and map these data items into points in some f-dimensional space (f is user defined), such that distances are preserved. It has the following attractive properties:

- It is as fast as it can be, since it is *linear* on the number of items. (No mapping algorithm can be better than linear, unless it operates on a sample of the database, which FastMap can do, too.) Specifically, it needs $\mathcal{O}(fN)$ distance calculations.
- It is dynamic: New data can be mapped after the original data have been mapped, without having to redo the mapping from the beginning. The mapping for a new item (or query) takes $2\ f$ distance calculations, that is, constant on the size of the database.
- The algorithm does not require the distance measure to be Euclidean. However, on non-Euclidean distances, we may have false dismissals, as we see later.

3 Introduction to ARGs

We adopted ARGs as the underlying image content representation and the editing distance between ARGs to express (dis-)similarity between images. The problem of retrieving images which are similar to a given query 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 the stored ARGs. Next, we

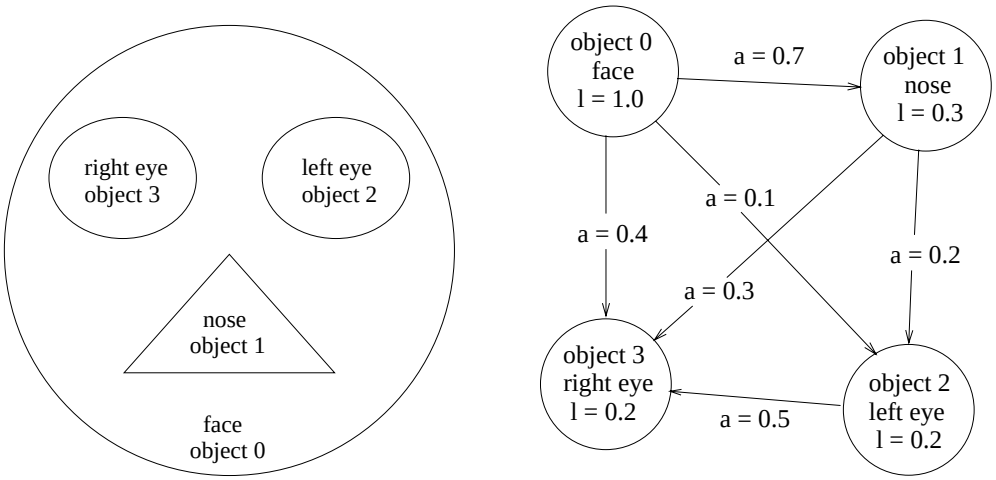

Figure 1: *Example image showing a sketch of a face (left) and its corresponding ARG (right).*

see (a) What types of attributes to keep, (b) How to define the editing distance between two ARGs and (c) How to compute it efficiently.

In an ARG, image objects or regions are represented by graph nodes and their relationships are represented by arcs (edges) between such nodes. Both nodes and arcs are labeled by attributes corresponding to properties of regions and relationships respectively. Figure 1 gives an example of an image (a mocked-up "human face", with face periphery, two eyes and a nose) and its corresponding ARG.

The specific attributes which are used in ARGs are derived from the raw image data. Typical attributes are e.g., statistical or textural values, geometric features of regions (e.g., size, roundness) or features specified in some transform domain (e.g., Fourier coefficients of shapes). Additional, higher-level attributes, such as class names or concepts, can easily be handled by ARGs, too. Typical features for the relationships (the edges of the ARG) are the distance, relative orientation etc., of the two involved regions.

Each region, in the above toy example, has only one attribute, the length (l) of its boundary. The relationship between any two regions has also one attribute, the angle (a) with the horizontal direction of the line connecting the centers of mass of these regions. Both attributes have been normalized in the range $[0, 1]$ by division with the size of the biggest region (face outline) and 360 respectively.

3.1 Definition of the ARG Editing Distance

Due to noise and distortions (which always exist in real-world images) a query and a model ARG cannot be exactly the same. Therefore, ARG matching has to be error-tolerant. Algorithms for error-tolerant ARG matching are referred to as *"error correcting"* graph matching algorithms [9, 37, 38]. These algorithms are based on the idea of transforming one ARG to the other following a sequence of ARG *edit operations*, called *"error corrections"*. These corrections have the form of node and arc insertion (deletion) for missing (extra) nodes and arcs, and node and arc substitution for matched nodes and arcs respectively. The definition of costs for the above graph edit operations depends on the application. The distance between two ARGs is defined as *the minimum cost taken over all sequences of operations that transform the one ARG to the other*. The smaller the distance between two ARGs the most similar they are.

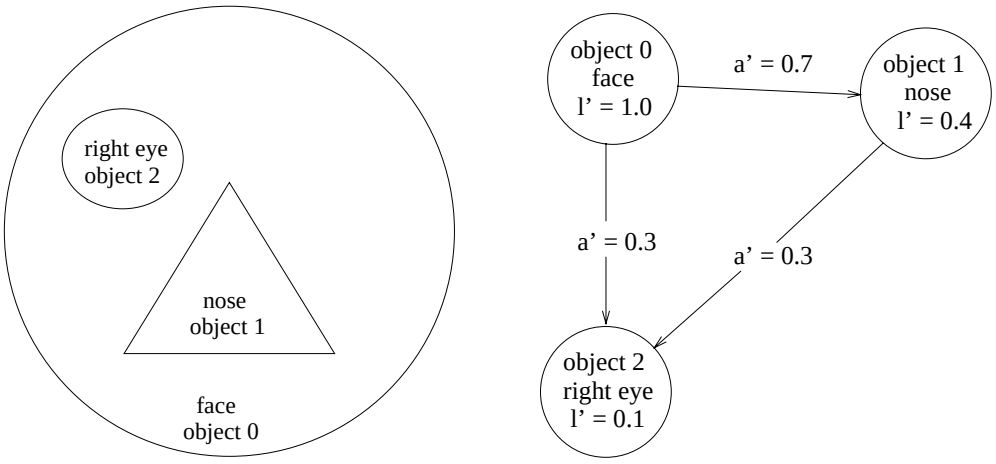

Figure 2: *Example query image (left) and its corresponding ARG (right).*

Representation	Substitution	Deletion or Insertion
Query Arc	$\|a' - a\|$	a'
Model Arc	$\|a - a'\|$	a
Query Node	$\|l' - l\|$	l'
Model Node	$\|l - l'\|$	l

Table 1: *Costs for arc and node insertion, deletion and substitution for the example matching.*

Next we show how to measure the editing distance between two ARGs: The query ARG of Figure 2 is matched with the image of Figure 1 (model).

The costs for node and arc insertion, deletion and substitution are defined in Table 1. There may exist weights associated with each such operation. For example, we may assign weights greater than 1 when nodes or edges with different labels are matched. However, since weights present no additional constraints on the methods which are discussed, we assume weights 1 for the rest of this paper.

An ARG matching algorithm would substitute (match): query region 0 with model region 0 with cost $|1.0 - 1.0| = 0$, query region 1 with model region 1 with cost $|0.4 - 0.3| = 0.1$, query region 2 with model region 3 with cost $|0.1 - 0.2| = 0.1$. The algorithm would also match the edges between the above regions with costs $|0.7 - 0.7| = 0$, $|0.3 - 0.4| = 0.1$ and $|0.3 - 0.3| = 0$. Object 2 in the model image is deleted and the cost of this operation is 0.2. Similarly, its edges are deleted with costs 0.5, 0.2 and 0.1. Hence, the total cost of matching is $0.1+0.1+0.1+0.5+0.2+0.1 = 1.1$. Notice that this is the least cost. For example, if we match query region 2 with region 2 in the model image the cost will become 1.8.

Intuitively, the more similar regions with similar relationships two images have, the more similar they are. By adding more attributes to the nodes and the edges of the above ARG representation we can enforce more rigorous matching.

In [2], the cost of matching would be 0.3 since, extra nodes and edges in the model image do not contribute to the cost of matching. This case of matching is referred to as "subset matching" or "subgraph error correcting isomorphism". Subset matching is applied when we want to find if the query regions exist in a model (stored) image. The cost of matching can be small even if the

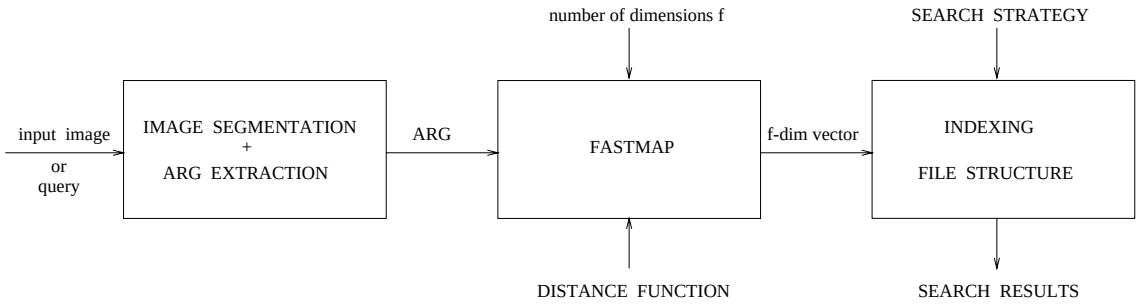

Figure 3: *ImageMap: Block diagram.*

images look dissimilar.

3.2 Speed of ARG Distance Computation

The methods for ARG matching referred to above find the optimal solution but they have exponential time (and space) complexity in the worst case. Approximate methods with lower time complexity do exist but they are not guaranteed to find the optimal solution. For example, Christmas, Kittler and Petrou proposed a method based on probabilistic relaxation [39] and Almohamad and Duffuaa proposed a method based on linear programming [40].

To speed-up retrievals, the stored ARGs must be indexed: Segupta and Boyer proposed a hierarchical organization of the stored model graphs [13]. The root graph consists of different distinct subgraphs of the model graphs which are matched with the query. The disadvantages of this method are: (a) Matching with the root graph may become very time consuming, (b) Assumes that the stored models are similar to each other, (c) Does not guarantee high recall, (d) It employees expensive preprocessing and (e) It is static, that is new models cannot be added (i.e., the entire preprocessing step has to be repeated).

Messmer and Bunke create a decision tree from the stored graphs [12]. A set of distorted copies for each graph is created and represented in the decision tree. The degree of distortion must be known in advance. However, the size of the decision tree increases exponentially with the size of the stored graphs and with the degree distortion. Due to its large space requirements, this method cannot be used in image databases. Addition work on ARG indexing assumes specialized cases of ARGs: For example, in [11] the graphs are not attributed and matching seems to work only for simplistic planar shapes. Our proposed method handles all these issues.

4 Proposed Method

ImageMap, maps each image to a point in an f-dimensional space. The mapping of ARGs to f-dimensional points is achieved through FastMap [36]. Figure 3 illustrates the above sequence of operations. Similarly, queries are mapped to points in the above f-dimensional space and the problem of IDB search is transformed into one of spatial search. To speed-up retrievals, the f-dimensional points are indexed using an R-tree. Below we discuss each one of the above processing steps separately.

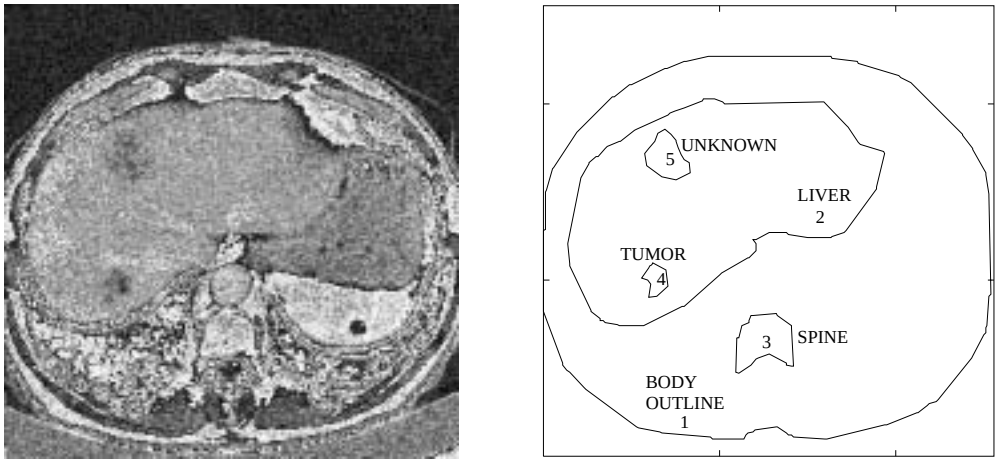

Figure 4: *Example of an original grey-level image (left) and its segmented form (right).*

4.1 Image Segmentation

All images are segmented into closed contours corresponding to dominant image objects or regions. However, image segmentation and labeling of the components are outside the scope of this paper. For our purposes, we assume that each image has been segmented manually (e.g., by tracing the contours of the regions of interest). Figure 4 shows an example of an original MRI image and of its segmented form. The contribution of our work is on the fast searching after the images and the queries have been segmented.

4.2 ARG Representation

Figure 5 shows the proposed ARG representation for medical MRI images of the abdomen such as the example image of Figure 4. Nodes correspond to regions and arcs correspond to relationships between regions. Both nodes and arcs are labeled by the attribute values of the region properties and the relationship properties, respectively. Angles are in degrees. In this work we used the following set of features:

Individual regions are described by 3 attributes, namely *Size* (s), computed as the size of the area of a region, *Roundness* (r), computed as the ratio of the smallest to the largest second moment, and *Orientation (o)*, defined as the angle between the horizontal direction and the axis of elongation. This is the axis of least second moment.

Spatial Relationships between regions are described by 2 attributes, namely *Distance* (d), computed as the minimum distance between their contours and *Relative Angle* (a), defined as the angle with the horizontal direction of the line connecting the centers of mass of the two regions.

It is straightforward to add more features as region or relationship attributes. Additional features that could be used include the average grey-level and texture values, moments or Fourier coefficients etc. as region descriptors; relative size, amount of overlapping or adjacency etc. can be also used to characterize the relationships between regions. In any case, our proposed method can handle *any* set of features.

However, we have strong reasons to use the above set of features:

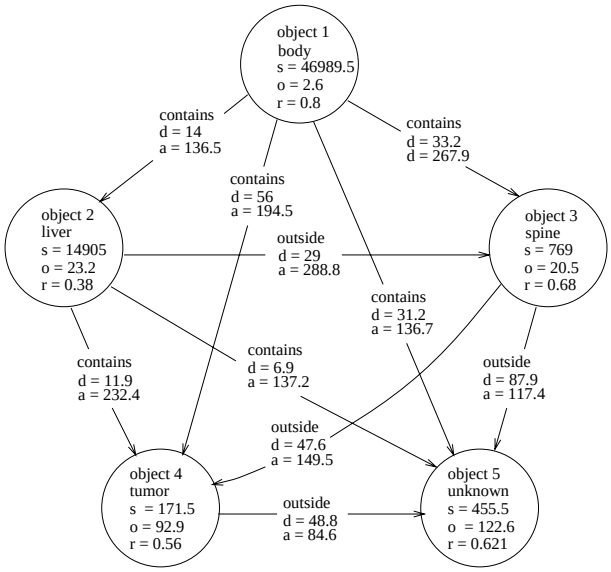

Figure 5: *Attributed Relational Graph (ARG) corresponding to the example image of Figure 4.*

- The features we used are very successful: We have used the same set of features in previous work [41, 25, 2] and we have discussed the results with experts. Our set of features is superset of the set of features used by other researchers (e.g., [26]).

- The derived representation is both scale and translation invariant (i.e., images translated or scaled with respect to each other result in the same representation). To achieve translation invariance, we take only relative positions. To achieve scale invariance, we normalize lengths and areas by dividing them respectively by the diameter and the area of the largest region. We also achieve rotation invariance, by registering all images to a standard orientation (e.g., the axis of elongation of the outline/largest region is made horizontal).

4.3 Mapping ARGs to f-dimensional Points

The FastMap algorithm accepts as input N ARGs, the Eshera and Fu distance function and f, the desired number of dimensions, and maps the above ARGs to N points in an f-dimensional space. As mentioned earlier, the complexity of this mapping is $\mathcal{O}(N f)$ distance computations. Notice that this operation could (and should) be done off-line. When an input image or a query is given, it is quickly mapped to a point into the above f-dimensional space requiring only $2f = \Theta(f)$ distance calculations. Notice that, in contrast to [2], each image is mapped to exactly one point.

4.4 Indexing - File Structure

Figure 6 demonstrates the proposed file structure of the data on the disk. Specifically, the file structure consists of the following parts:

- The spatial access method, holding an f-dimensional vector for each stored image (ARG). We used R-trees solely because of availability; *any* spatial access method would do, like, e.g., R*-trees and X-trees. In fact, a faster SAM would only make our approach work even faster!

- The *"ARG file"*. This is a file holding the ARGs. Each record in this file consists of (a) An identifier (e.g., the image file name) corresponding to the image from which the ARG has

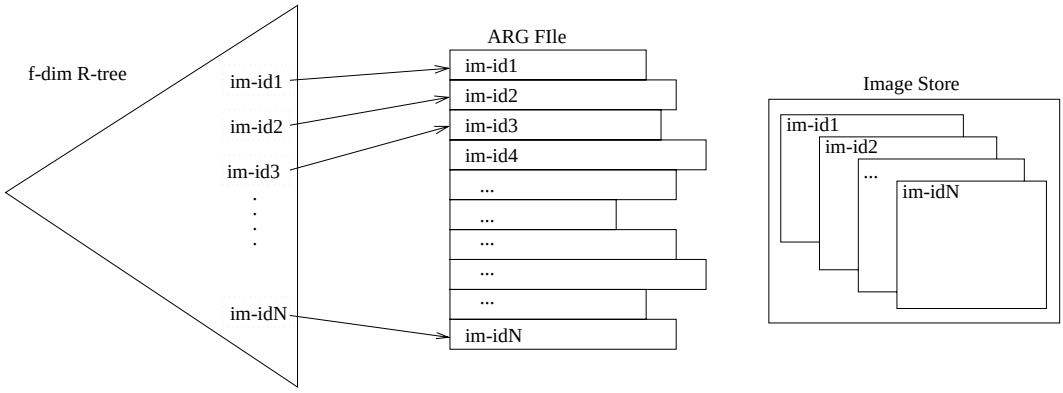

Figure 6: *File structure.*

been derived and (b) The features of each region together with its relationships with the other regions.

- The *"Image store"* holding the original image files. For faster display, we have also kept the segmented forms of all images.

4.5 Search Strategy

The user specifies a query image and a tolerance t, and asks for all the images within that tolerance. The ARG of the query is computed first. Then, the f-dimensional vector of the above ARG is derived. All vectors within tolerance t are retrieved from the R-tree. As we show in Section 5, the R-tree may return false alarms (i.e., not qualifying images). A post-processing step is required to clean-up the false alarms. The generic search algorithm is as follows:

R-tree search: Issue a range query on the 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 ARG 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.

Nearest-neighbor queries can also be answered, using the algorithm in [42]. The idea is to use the R-tree to find some near neighbors in the f-space and to ask a range query with a carefully chosen radius, being ready to clean-up potential false alarms.

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 specific ARG properties. The weights could even be adjusted on-the-fly by the user. Since a weighted ARG 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 a large dataset of segmented images has to be used. In this work we used the following datasets:

- REAL: consisting of 124 real MRI images manually segmented containing between 2 and 6 regions.

- SYNTHETIC: To test scalability we generated 10,000 images as in [2] by random small perturbations on 50 original MRI images which are manually segmented. All images contain 4 regions. The images require 12.5 Mbytes (uncompressed) for storage and their file of ARGs 7.6 Mbytes. The space for the implementation of the R-tree is 278.5 Kbytes for $f = 2$ dimensions, 405.5 Kbytes for $f = 3$ and 618.5 Kbytes for $f = 5$.

We implemented our method in ANSI C and C++ under UNIX^{TM} and we run the experiments on a dedicated SUN/20.

Each point in our plots is the average over 50 characteristic queries. We carried out several groups of experiments. The experiments were designed to:

- Illustrate the *superiority* of our method over sequential scan searching. We studied the search time for various values of the tolerance t and of the dimensionality f on the SYNTHETIC dataset. We show that our method is up to 3 orders of magnitude faster than sequential scanning and scales-up well for large databases. The times reported correspond to the elapsed time computed using the `time` system call of UNIX. These are times for database search; times for image display are not reported, since they are common to all methods. Notice that 99% of the response time was due to ARG distance calculations, while only 1% was dedicated to disk accesses and other system delays.

- Study the *accuracy* of our method. We show that our method produces almost always the same responses that the sequential scan method produces, but much faster. To measure the accuracy of the method, we used *recall*, that is, the number of retrieved images as a percentage of the number of actually qualifying images, assuming that the sequential algorithm yields the correct answers ("ground-truth"). We did not perform experiments to judge the "goodness" of the set of ARG features we used, since these features have been successfully used before [41, 25, 2].

5.1 Response Time

Figure 7 shows the elapsed (wall-clock) average search time plotted against the tolerance t for $f =$ 2, 3 and 5 dimensions. The search time for our method includes the R-tree search time plus the clean-up time. For $t \leq 2$, up to 20 images are retrieved on the average. Notice that, our method retrieves always the most similar images (best matches).

Sequential scan takes the same time, for any tolerance, as expected. For our proposed method, the effort increases with the tolerance. However, even for tolerance $t = 2$ (which retrieves roughly 20 images), the response time is significantly better than sequential scanning. For smaller tolerance, say $t = 0.5$ (which retrieves 2 images on the average) the proposed method takes from 0.9 seconds ($f = 5$) to 50 seconds ($f = 2$), which is, 1-3 orders of magnitude smaller than sequential scanning. For $t = 1.1$ and $f = 5$ the method retrieves 5 images; the response time in this case is approximately 20 seconds, that is, 67 times faster than sequential scanning.

The effect of f should be noted: Higher f leads to more selective filtering, and thus, fewer false alarms and less effort on clean-up.

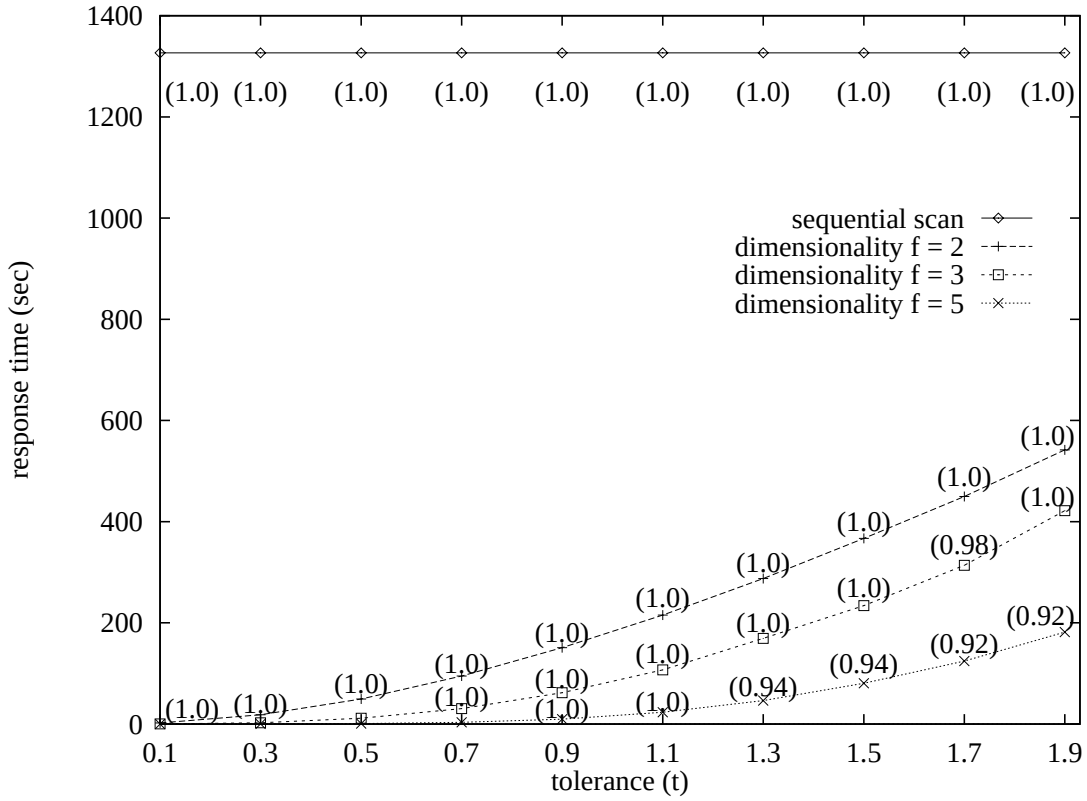

Figure 7: *Average retrieval response time on the SYNTHETIC dataset, corresponding to indexed and sequential scan searching, as a function of the tolerance t, for dimensionality f=2, 3 and 5. The labels denote the average value of recall.*

5.2 False Dismissals

The labels in Figure 7 denote the average value of recall. Notice that the recall is always 1 (i.e., 100% accuracy) for sequential scanning. For the proposed method the recall is always above 0.92. If accuracy is crucial, then we would choose $f = 3$; otherwise, we would choose $f = 5$ (response times are faster). Our method retrieves the 5 best matches ($t = 1.1$) with 100% accuracy and the 20 best matches ($t = 2$) with accuracy 92%-100%, depending on the number f of dimensions.

5.3 Scale-up

For larger datasets, the above experimental results scale accordingly. Figure 8 shows the response time for both sequential scanning and our method, as a function of the database size (number of stored images). Notice that our method is consistently faster than sequential scanning. The performance gap between the two methods widens as the database grows. Therefore, the proposed method becomes increasingly attractive for larger databases. The labels in the plot of Figure 8 denote the average value of recall. In all cases, the recall was above 94%, and typically 100%, in the majority of cases.

5.4 Visualization

Figure 9 shows the diagram for $f = 2$ dimensions on the REAL dataset, that is, 124 real tomographic images, manually segmented. Notice that, there are 5 clusters corresponding to images

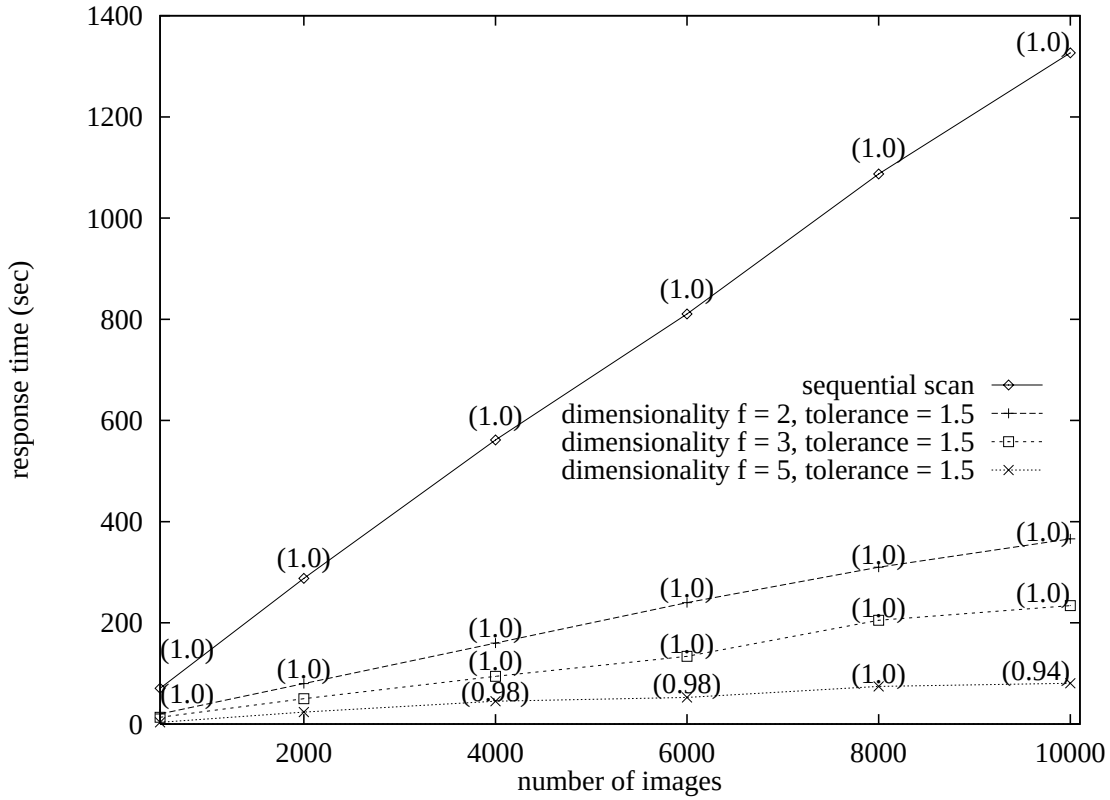

Figure 8: *Average retrieval response time on the SYNTHETIC dataset, corresponding to indexed and sequentical scan seaching, as a function of the database size, for tolerance $t = 1.5$ and dimensionality f 2, 3 and 5. The labels denote the average value of recall.*

with 2, 3, 4, 5, and 6 regions respectively. This is because we designed the distance function to respond to missing or extra regions. Missing (extra) regions increase the ARG distance. Therefore, images differing in the number of regions appear far apart on the target "space" while, images with the same number of regions tend to cluster together.

The diagram of Figure 9 can be found especially useful for browsing the contents of the database. Once a query is mapped to a point on the above diagram, one could search manually for similar images by browsing images mapped in the neighborhood of the query. Queries address clusters with the same number of regions thus narrowing down the search approximately to the one fifth of the database size.

Figure 10 demonstrates a characteristic example of a query image (top left) specifying 5 regions, on the REAL dataset. The arrow in Figure 9 points to the projection of the query image on the 2-dimensional space. Notice that since the query has 5 regions, it was correctly mapped to a point close to the cluster of other 5-region images. Its three nearest neighbors in the above 2-dimensional diagram are indicated with "$5*$", and are shown in Figure 10 (top right, and both bottom images).

6 Conclusions

We introduce ImageMap, a method for handling similarity searching in Image DataBases (IDBs) which has the following desirable properties:

- It proved to be up to 3 orders of magnitude faster than sequential scanning.

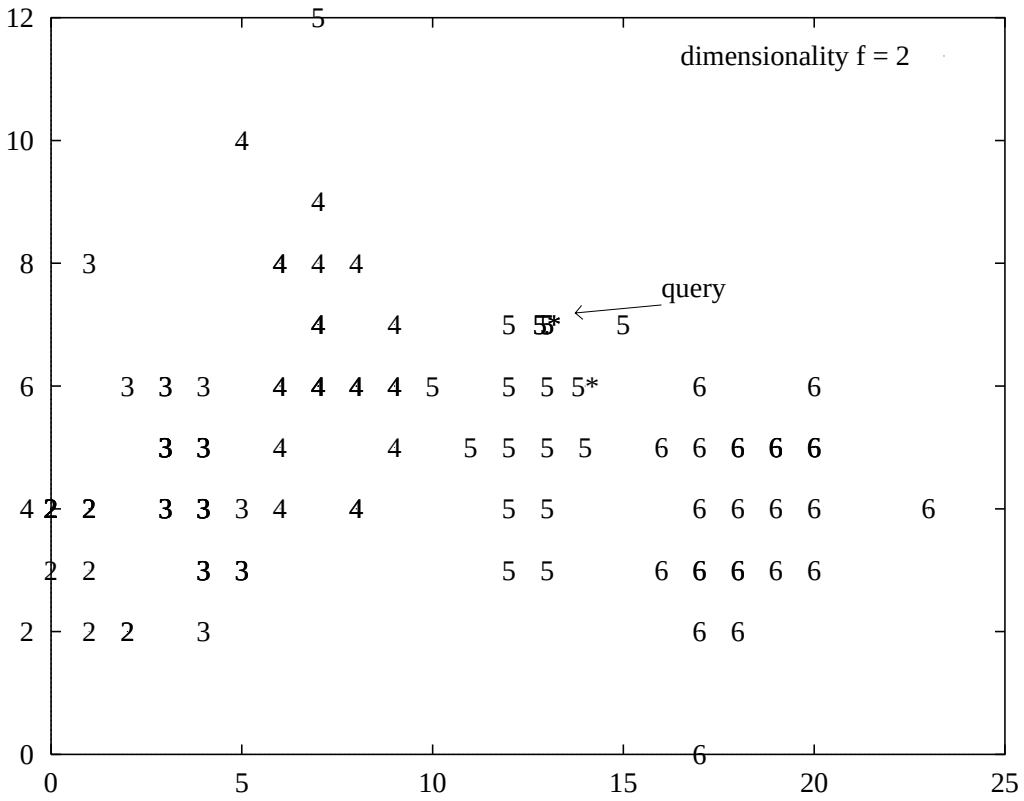

Figure 9: *Diagram for 124 real MRI images (REAL dataset). There are 5 clusters corresponding to images with 2, 3, 4, 5 and 6 regions respectively. The labels denote number of regions in images. The arrow denotes the projection of the query of Figure 9. Its 3 nearest neighbors are denoted by "5∗".*

- It introduced no false alarms and few, or no false dismissals (the recall was greater than 0.92 and usually 1.00).
- The method scales-up well for large databases (i.e., the performance gap between our method and sequential scanning widens as the database grows).
- It allows visualization, browsing and data-mining on image database contents.
- It incorporates a more efficient method to compute the ARG editing distance.

ImageMap is also a method for the indexing of stored ARGs. Methods such as [13, 12] require expensive preprocessing, perform well only on homogeneous data, are static, do not guarantee high recall or have prohibitively large space overhead. The proposed method handles all these issues. In our previous work [2], we did not allow for missing or extra regions and we required that at least some of the regions are labeled and present in every image and query. In the current work, we relax these restrictions. Finally, the proposed method generalizes representations used by other researchers and provides index support to existing methodologies such as [26].

With respect to future work, a very promising direction is the study of data-mining algorithms [43, 44] on the point-transformed set of images, to detect regularities and patterns, as well as to detect correlations with demographic data. Another promising direction is the use of more recent indexing structures, such as the R*-tree [30] or the X-tree [31], which promise faster search times for smaller space overhead.

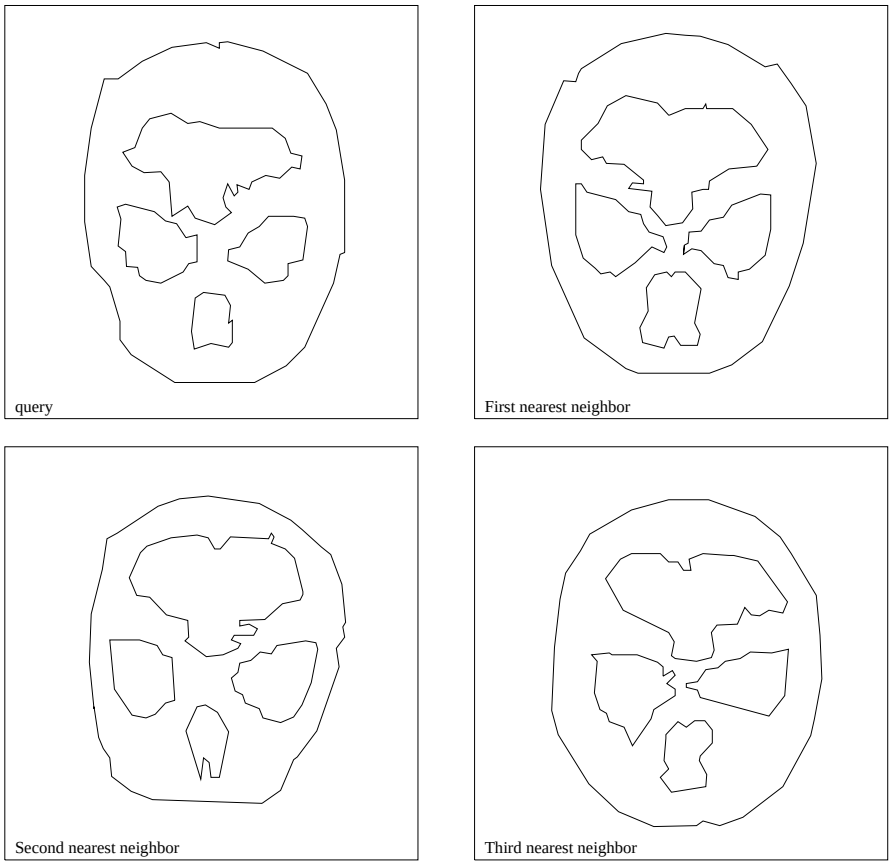

Figure 10: *Example of a query image (top left) and of its 3 nearest neighbors.*

A Appendix: The Eshera-Fu Distance Computation Algorithm

Here we discuss the Eshera-Fu ARG matching algorithm [9]. In the next section we propose a more efficient implementation of it.

In the following, the ARG at the left of Figure 11 (query ARG) is compared with the ARG at the right (model or reference ARG).

The algorithm works by decomposing each ARG into a set of "Basic Attributed Relational Graphs" (BARGs). Each BARG has the form of an one level tree consisting of a root node corresponding to an ARG node, the arcs emanating from it and the nodes on which these arcs terminate (terminal or leaf nodes). Figure 12 illustrates the BARGs which are produced from the example ARGs.

The algorithm creates the state-space tree of Figure 13. Each state (tree node) corresponds to a matching of a pair of subgraphs from the two input ARGs. Each state is labeled by a symbol s, numbered 0 through 11 (i.e., the number of states which are produced). The root of the state-space tree is labeled by s_0 denoting the matching of (initially) empty subgraphs. A transition from a state to another corresponds to the embedding (matching) of an unmatched pair of BARGs into the already matched subgraphs. The edit operations for each embedding are recorded and their costs are accumulated. Each state in Figure 13 is also labeled with the pairs of the embedded BARGs. Each such pair contains the index of the query BARG, followed by the index of the model BARG.

The algorithm terminates when both ARGs have been reconstructed. A path from the root to a leaf of the state-space tree denotes a sequence of operations that transforms the query into the

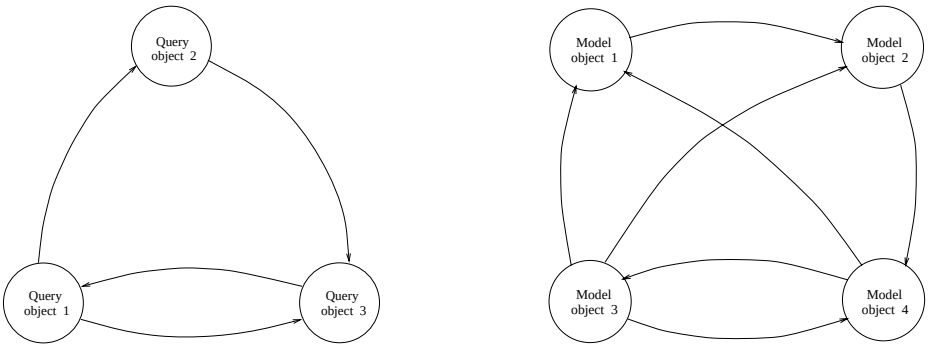

Figure 11: *Example of a query (left) and of a reference ARG (right).*

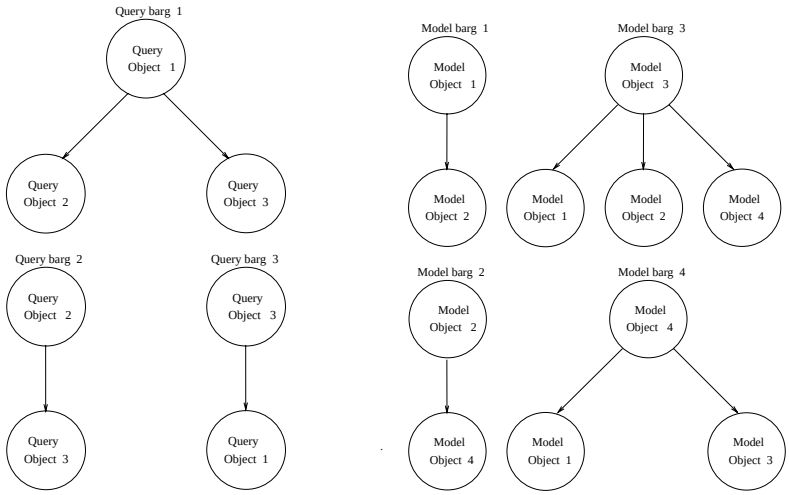

Figure 12: *Basic Attributed Relational Graphs (BARGs) produced from the query (left) and the model ARG (right).*

model ARG (e.g., the path from s_0 to s_7 in Figure 13). The path with the minimum cost yields the cost of ARG matching. In matching the ARGs of Figure 11, region 1 in the query substitutes (matches) region 3 in the model ARG, region 3 substitutes region 4 and region 2 substitutes region 2. Object 1 in the model ARG and its related arcs are deleted.

For a complete analysis of the complexity the interested reader is referred to [9]. Assuming that the degree δ of the ARGs which are matched is small enough compared to the number of nodes M, N in the two ARGs, the time complexity of the algorithm is $\mathcal{O}\left(M^2N^2(M+N)\right)$.

B Appendix: Our ARG Distance Computation Algorithm

In the original algorithm, prior to computing the minimum distance, all possible solutions must be computed. We propose the following optimization: We don't extend paths yielding cost greater than the *upper bound* that is, the cost of the best solution found so far. To achieve an early solution (upper bound), the state-space tree is expanded in depth-first fashion. State s_7 in Figure 13 is a solution which sets an upper bound; states s_9 and s_{11} need not be expanded. Once all possible

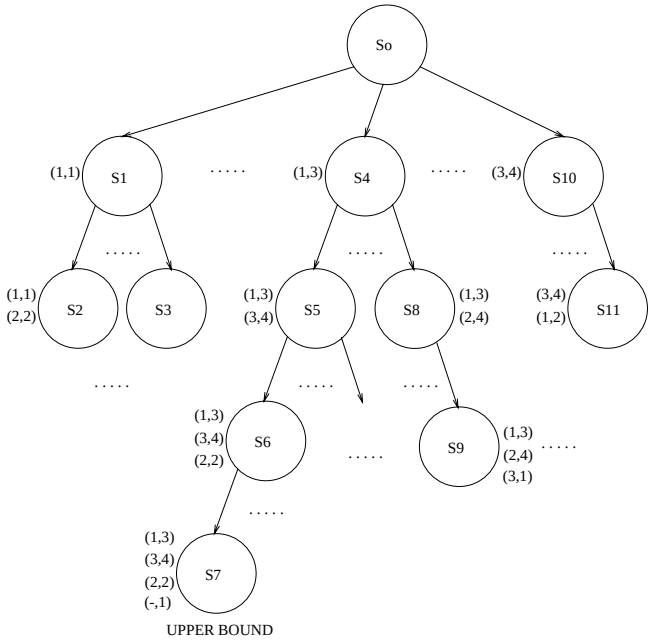

Figure 13: *State-space tree of ARG matching. Each node corresponds to a state and each state is labeled with the indices of the embedded BARGs.*

Attribute Representation		Substitution	Deletion or Insertion
Query Arc:	$(Qa_1, Qa_2, \ldots Qa_n)$	$\sum_{i=1}^{n} \lvert Qa_i - Da_i \rvert$	$\sum_{i=1}^{n} Qa_i$
Data Arc:	$(Da_1, Da_2, \ldots Da_n)$	$\sum_{i=1}^{n} \lvert Qa_i - Da_i \rvert$	$\sum_{i=1}^{n} Da_i$
Query Node:	$(Qb_1, Qb_2, \ldots Qb_m)$	$\sum_{j=1}^{m} \lvert Qb_j - Db_j \rvert$	$\sum_{j=1}^{m} Qb_j$
Data Node:	$(Db_1, Db_2, \ldots Db_m)$	$\sum_{j=1}^{m} \lvert Qb_j - Db_j \rvert$	$\sum_{j=1}^{m} Db_j$

Table 2: *Costs for arc and node insertion, deletion and substitution.*

paths have been explored, the minimum total cost corresponds to the last upper bound (e.g., the cost of state s_7 in our example).

The heart of the method is an algorithm for the matching of BARGs. The original algorithm doesn't handle this problem assuming the straightforward way to implement matching by finding the minimum cost between all possible configurations of nodes, which is exponential in the worst case (i.e., for large δ); hence the complexity of ARG matching becomes exponential. Our proposed method is polynomial: The problem of finding the best mapping between two BARGs is modeled as an *assignment* problem (i.e., each root-arc-terminal triple of a BARG is considered to be a node of a bipartite graph) which is solved using the Hungarian method [45] in $\mathcal{O}(\delta^3)$ time.

In matching two BARGs, extra (missing) arcs and nodes are deleted (inserted). Matching two nodes or arcs is equivalent to substitution. The costs of node and arc insertion, deletion and substitution are defined in Table 2. Symbols Qa, Da denote arc attributes, symbols Qb, Db denote node attributes while, n, m denote number of arc and node attributes respectively. The attributes we used in this work are given in Subsection 4.2. Notice that, substituting or inserting (deleting) a node or an arc, is equivalent to substituting or deleting (inserting) a node or arc in the other BARG with the same cost.

References

[1] S. C. Orphanoudakis, C. Chronaki, and D. Vamvaka. I^2Cnet: Content-Based Similarity Search in Geographically Distributed Repositories of Medical Images. *Computerized Medical Imaging and Graphics*, 20(4):193–207, 1996.

[2] Euripides G.M. Petrakis and Christos Faloutsos. Similarity Searching in Medical Image Databases. *IEEE Transactions on Knowledge and Data Engineering*, 9(3), May/June 1997.

[3] Jian Kang Wu and Arxot Desai Narasimhalu. Identifying Faces Using Multiple Retrievals. *IEEE Multimedia*, 1(2):20–38, 1994.

[4] Nalini K. Ratha, Kalle Karu, Shaoyun Chen, and Anil K. Jain. A Real Time Matching System for Large Fingerprint Databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):799–813, August 1996.

[5] Babu M. Mehtre, Mohan S. Kankhanhalli, and Wing Foon Lee. Shape Measures for Content Based Image Retrieval: A Comparison. *Information Processing and Management*, 33(3):319–337, 1997.

[6] 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.

[7] M. De Marsicoi, L. Cinque, and S. Levialdi. Indexing Pictorial Documents by Their Content: A Survey of Current Techniques. *Image and Vision Computing*, 15(1):119–141, 1997.

[8] Venkat N. Gudivada and Vijay V. Raghavan. Modeling and Retrieving Images by Content. *Information Processing and Management*, 33(4):427–452, 1997.

[9] 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.

[10] Heikki Kalviainen and Erkki Oja. Comparisons of Attributed Graph Matching Algorithms for Computer Vision. In *Proceeding of STeP-90 Finish Artificial Intelligence Symposium*, pages 354–368, University of Oulu, June 1990. http://www.lut.fi/˜kalviai/bibliography.html.

[11] H. Sossa and R. Horaud. Model Indexing: The Graph Hashing Approach. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 811–815, 1992.

[12] B. T. Messmer and H. Bunke. Fast Error-Correcting Graph Isomorphism Based on Model Precompilation. Technical Report IAM-96-012, University of Bern, Switzerland, September 1996. http://www.iam.unibe.ch/˜fkiwww/publications/index.html#technicalreports.

[13] Kuntal Segupta and Kim L. Boyer. Organizing Large Structural Modelbases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):321–332, April 1995.

[14] Ramesh Jain, Alex Pentlant, and Dragutin Petkovic. NSF-ARPA Workshop on Visual Information Management Systems, June 1995. http://www.virage.com/vim/vimsreport95.html.

[15] Amarnath Gupta and Ramesh Jain. Visual Information Retrieval. *Communications of the ACM*, 40(5):71–79, May 1997.

[16] M. Flickner et. all. Query By Image and Video Content: The QBIC System. *Computer*, 28(9):23–32, September 1995.

[17] A. Pentland, R. W. Picard, and A. Sclaroff. Photobook: Content Based Manipulation of Image Databases. *International Journal of Computer Vision*, 18(3):233–254, 1996.

[18] Rajiv Mehrota and James E. Gary. Similar-Shape Retrieval in Shape Data Management. *Computer*, 28(9):57–62, September 1995.

[19] Boon-Lock Yeo and Minerva M. Yeung. Motion Recovery for Video Classification. *ACM Transactions on Information Systems*, 13(4):408–439, October 1995.

[20] Nevenka Dimitrova and Forouzan Golshani. Motion Recovery for Video Content Classification. *ACM Transactions on Information Systems*, 13(4):408–439, October 1995.

[21] Alberto Del Bimbo, Enrico Vicario, and Daniele Zingoni. Symbolic Description and Visual Querying of Image Sequences Using Spatio - Temporal Logic. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):609–621, August 1995.

[22] 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.

[23] S.-K. Chang, E. Jungert, and G. Tortora, editors. *Intelligent Image DataBase Systems*. World Scientific, 1996.

[24] John R. Smith and Shih-Fu Chang. VisualSEEk: A Fully Automated Content Based Image Query System. In *Proceedings of ACM Multimedia Conference*, Boston Ma., November 1996.

[25] 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.

[26] Venkat N. Gudivada and Vijay V. Raghavan. Design and Evaluation of Algorithms for Image Retrieval by Spatial Similarity. *ACM Transactions on Information Systems*, 13(2):115–144, 1995.

[27] 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.

[28] C. Faloutsos and S. Roseman. Fractals for Secondary Key Retrieval. *Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 247–252, March 1989. also available as UMIACS-TR-89-47 and CS-TR-2242.

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

[30] Nobert Beckmann, Hans-Peter Kriegel, Ralf Schneider, 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.

[31] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree : An Index Structure for High-Dimensional Data. In *Proceedings of the 22rd VLDB Conference*, pages 28–39, 1996.

[32] Norio Katayama and Shinichi Satah. The SR-Tree: An Index Structure for High Dimensional Nearest Neighbor Queries. In *Proceedings of ACM SIGMOD*, pages 369–380, Tuscon, Arizona, May 1997.

[33] P. N. Yianilos. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.

[34] Sergey Brin. Near Neighbor Search in Large Metric Spaces. In *Proceedings of the 21st VLDB Conference*, pages 574–584, Zurich, Switzerland, 1995.

[35] Tolga Bozkaya and Meral Ozoyoglu. Distance-Based Indexing for High-Dimensional Metric Spaces. In *Proceedings of ACM SIGMOD*, 1997.

[36] C. Faloutsos and K.-I. Lin. FastMap: A Fast Algorithm for Indexing, Data Mining and Visualization of Traditional and Multimedia Datasets. In *SIGMOD RECORD, Proceedings of the 95 ACM SIGMOD International Conference on Management of Data*, pages 163–174, San Jose, California, June 1995.

[37] Alberto Sanfeliou and King-Sun Fu. A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. *IEEE Transactions on Systems Man and Cybernetics*, SMC-13(3):353–362, 1983.

[38] E. K. Wong. Three Dimensional Object Recognition by Attributed Graphs. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition - Theory and Applications*, pages 381–414. World Scientific, 1990.

[39] William J. Christmas, Josef Kittler, and Maria Petrou. Structural Matching in Computer Vision Using Probabilistic Relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):749–764, August 1995.

[40] H. A. Almohamad and S. O. Duffuaa. A Linear Programming Approach for the Weighted Graph Matching Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:522–525, 1993.

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

[42] Flip Korn, Nikolaos Sidiropoulos, Christos Faloutsos, Eliot Siegel, and Zenon Protopapas. Fast Nearest Neighbor Search in Medical Image DataBases. In *Proceedings of Internation Conference on Very Large DataBases (VLDB)*, pages 215–226, Bombay, India, September 1996.

[43] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining Association Rules Between Sets of Items in Large Databases. *Proc. ACM SIGMOD*, pages 207–216, May 1993.

[44] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *Proc. of VLDB Conf.*, pages 487–499, September 1994.

[45] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, chapter 11, pages 247–255. Engelwood Cliffs: Prentice Hall, 1982.

Biographies

Euripides Petrakis received the BSc in physics from the National University of Athens, Greece in 1985 and the PhD degree in computer science from the University of Crete, Greece in 1993. He is currently an assistant professor at the Technical University of Crete. His research interests include image and video databases, access methods for spatial and geographic data, medical image databases and computer vision. He is a member of the IEEE Computer Society.

Christos Faloutsos received the BSc in electrical engineering from the National Technical University of Athens, Greece, and the MSc and PhD degrees in computer science from the the University of Toronto, Ontario, Canada. He is currently an associate professor at the University of Maryland, College Park. He received the U.S. National Science Foundation Presidential Young Investigator award in 1989. His research interests include access methods for text, spatial and geographic data, declustering, medical image databases and multimedia databases. He is a member of the IEEE Computer Society.

King-Ip (David) Lin received the BSc degree in computer studies from the University of Hong Kong in 1990 and the M.S. and PhD degree in computer science in 1993 and 1996 respectively from the University of Maryland at College Park. He is currently an assistant professor at the University of Memphis. His research interests include database systems, data mining, indexing and searching for multimedia data. He is a member of the ACM and the IEEE Computer Society.