ImageMap: An Image Indexing Method Based on Spatial Similarity

Euripides G.M. Petrakis
Dept. of Electronic and Computer Engineering
Technical University of Crete
Chanea, Greece
e-mail: `petrakis@ced.tuc.gr`

Christos Faloutsos [*]
Dept. of Computer Science
Carnegie Mellon University
e-mail: `christos@cs.cmu.edu`

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

July 30, 2001

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, attributed relational graph, editing distance, image indexing.

[*]This research was partially funded by the National Science Foundation under Grants No. IRI-9625428 and DMS-9873442, by the National Science Foundation, ARPA and NASA under NSF Cooperative Agreement No. IRI-9411299 and by DARPA/ITO through Order F463, issued by ESC/ENS under contract N66001-97-C-851. Additional funding was provided by donations from NEC and Intel. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the United States Government.

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 [1, 2, 3]. 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 [4], image content is given primarily in terms of properties of color and texture. The QBIC system of IBM [5] utilizes a retrieval method for queries on color, texture and shape. The Photobook [6], the system developed at the MIT Media Lab, supports queries by image content in conjunction with text queries. 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. In this work we tackle this problem. Specifically:

- We have a collection of N images. We used medical images as in Figure 4(left), but our method can handle any image type.
- Each image has been segmented (automatically or manually) to one or more objects or regions.
- We are also given a distance function between two images.
- The queries are "by example": The user specifies a desirable image (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. For two images to be similar, they must contain similar objects (regions in general) in similar spatial relationships.

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.
- 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 ARG editing distance

function on ARGs [7], a very general among other known distance functions. In this work, we used the distance function of [8], a more efficient implementation of the Eshera-Fu [9] distance function.

- We introduce a generic indexing technique to ARGs. Existing methods assume special kinds of ARGs [10], require prohibitively large space for storage [11], are static (i.e., they cannot handle insertions or deletions), do not guarantee high accuracy and assume that the images are similar [12]. Our proposed approach handles all these issues.

- This work completes and extends previous work by the authors [10].

The rest of this paper is organized as follows: A review of related work 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.

2 Related Work

2D strings [13] are examples of work focusing mainly on spatial relationships. VisualSEEk [14] 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) [15].

Attributed Relational Graphs (ARGs) are the most general representations of image content. Among other existing retrieval methods (e.g., [13, 15, 16]) are the most accurate, achieving higher precision and recall than any other method [17] but, require exponential time (or space) for matching [7]. Approximate ARG matching 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 [18] and Almohamad and Duffuaa proposed a method based on linear programming [19].

The method by Gudivada and Raghavan [16] emphasizes on special types of image properties (e.g., relative orientation between objects) for the representation of image content. The method by El-Kwae and Kabuka [20] is a substantial extension of the previous method to capture more types of spatial relationships. Both methods are invariant to image translation, scaling and rotation and guarantee polynomial time complexity for image matching. These methods assume that an association between objects in a query and a database image is given and they compute the cost of this association by taking the differences of the relationships between all pairs of matched objects in two images. ARG matching on the other hand, makes no assumption and computes the best association between objects in the two images by taking also their interrelationships into account. Both these methods assume sequential search of the entire IDB.

To speed-up retrieval response times the stored ARGs are indexed. In our previous work [10], 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.

The method by El-Kwae and Kabuka [21] is a multilevel indexing technique based on bit signatures. In Papadias, Mamoulis and Delis [22], the problem of answering queries specifying structural image content, is formulated as a Multiple Constraint Satisfaction (MCS) problem (i.e., each binary relation is considered as a separate constraint). Both, the database images and the queries are mapped to regions in a multidimensional space and are indexed by R-trees. Such queries are viewed as multi-way spatial joins, where spatial joins correspond to join predicates. The novelty of this approach is that it integrates MCS issues with SAMs along with query optimization issues (in an attempt to avoid the high computational complexity of the large number of spatial joins which are generated by each query). Except of its high complexity, this method treats only special types of relationships and object properties and it assumes all the images and the queries are at a known scale and orientation. In addition it cannot handle image translations.

Segupta and Boyer [12] proposed a hierarchical organization of graphs. The root graph consists of different distinct subgraphs derived from all the stored graphs. However, the matching of a query with the root graph may become very time consuming. In addition, the method is approximate (i.e., may miss an actual match). The method by Messmer and Bunke [11] relies on the idea of computing distorted copies of the stored graphs which are represented in a decision tree. The degree of distortion must be known in advance. The size of the decision tree, increases exponentially with the size of the stored graphs and with the degree of distortion. Due to its large space requirements, this method cannot be used in image databases. The methods referred to above require an expensive preprocessing step for building a tree index structure, work well for special forms of graphs (i.e., require that the stored graphs and the queries are similar and are not fully connected), they have not been tested on large datasets storing e.g., thousands of graphs and, finally, they are static (i.e., they cannot handle insertions or deletions of database images). Their emphasis is on the design of efficient algorithms for computing the error correcting isomorphism on graphs rather than on image retrieval by content.

3 Attributed Relational Graphs (ARGs)

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.

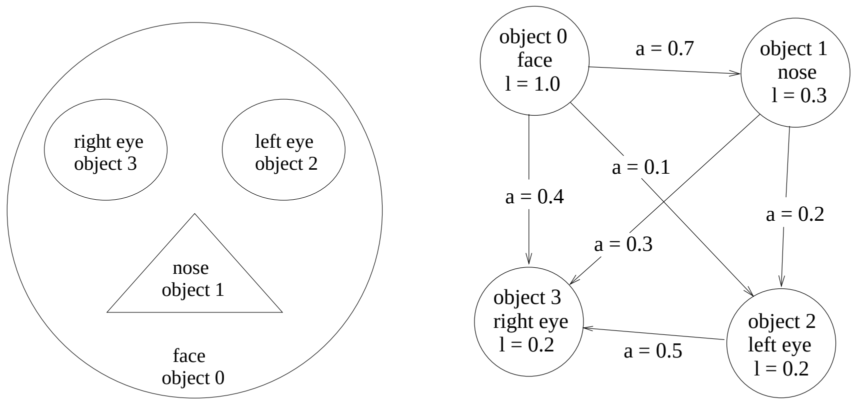

Figure 1: Example image showing a sketch of a face (left) and its corresponding ARG (right).

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 ARG Editing Distance

The editing distance between two ARGs is defined as *the minimum cost taken over all sequences of operations ("error corrections") that transform the one ARG to the other* [7]. The smaller the distance between two ARGs the most similar they are.

Next, 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. If more than one attributes are used in nodes or arcs, as it is actually the case in this work, their costs are added. There may exist weights associated with each such operation (e.g., we may assign weights greater than 1 when nodes or edges with different labels are matched). We assume weights 1 for the rest of this paper.

The computation of the distance between two ARGs involves not only finding a sequence of error correction that transform the one ARG to the other, but also finding the one that yields the minimum total cost. In the above example, an ARG matching algorithm would substitute (match): query region 0 with model

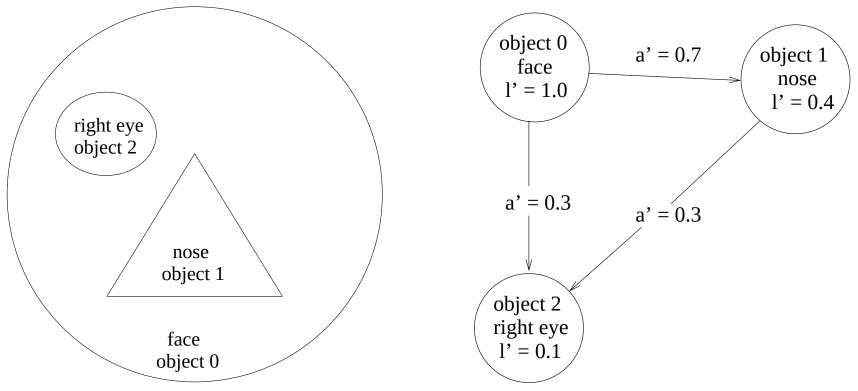

Figure 2: Example query image (left) and its corresponding ARG (right).

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

Table 1: Costs for arc and node insertion, deletion and substitution.

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. The total cost of matching is $0.1 + 0.1 + 0.1 + 0.2 + 0.5 + 0.2 + 0.1 = 1.3$. 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.

In our previous work [10], we did not allow for missing or extra regions in query (model) images and we did not allow for insertions and deletions. In [10], 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 images look dissimilar. In [10], we also required that at least some of the regions are labeled by an expert and present in every image. Comparisons between images with different labeled objects (e.g., misclassified objects) are not allowed. In the current work, we relax these restrictions.

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 [23]. 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.

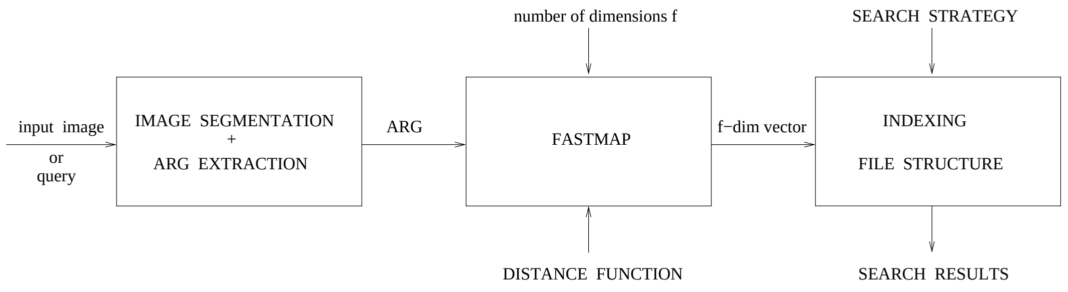

Figure 3: ImageMap: Block diagram.

4.1 Image Segmentation

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

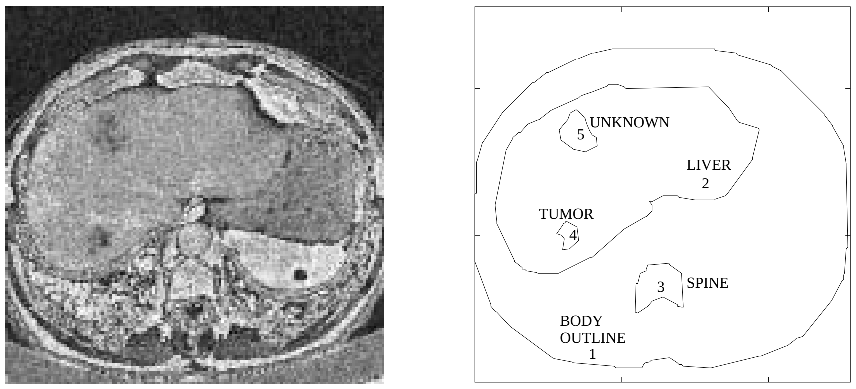

Figure 4: Example of an original grey-level image (left) and its segmented form (right).

4.2 ARG Representation

Figure 5 shows the proposed ARG representation for medical MRI images such as the example image of Figure 4. Nodes correspond to regions and arcs correspond to relationships between regions. The arcs are directed from the outer to the contained region but their direction also depends on object labels (e.g., arcs between tumor and unknown objects). Arcs between objects with the same label (e.g., tumors) could be bidirectional (this case is not present in Figure 5). Both nodes and arcs are labeled by the attribute values of the region properties and the relationship properties, respectively. 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. (i.e., axis of least second moment). Angles are in degrees.

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 (e.g., average grey-level and texture values, moments or Fourier coefficients etc., as region descriptors; relative size, amount of overlapping or adjacency for relationships etc.). In any case, our proposed method can handle *any* set of features. 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 - FastMap

FastMap [23] is an algorithm that takes in N ARGs, the ARG (Eshera-Fu) distance function and f, the desired number of dimensions, and maps the above ARGs to N points in an f-dimensional space such that distances are preserved. It has the following attractive properties:

- It needs only $\mathcal{O}(fN)$ distance calculations. This operation could (and should) be done off-line.
- It is dynamic: New data (i.e., an input image or a query) can be mapped after the original data have been mapped. The mapping for a new item takes $2f = \Theta(f)$ distance calculations.

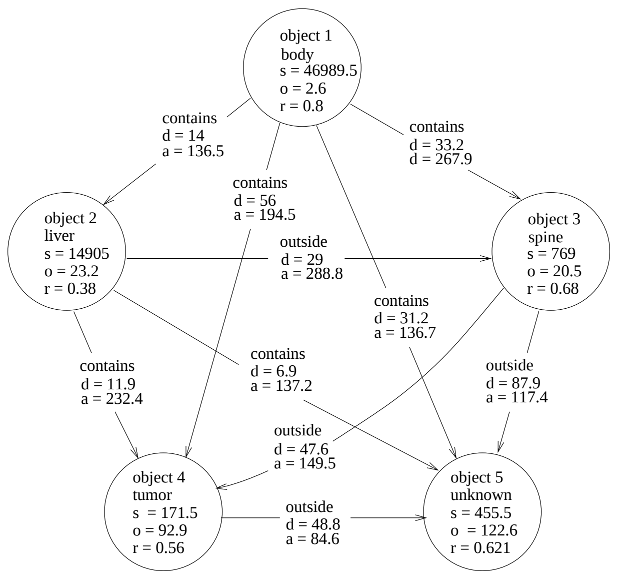

Figure 5: Attributed Relational Graph (ARG) corresponding to the example image of Figure 4.

4.4 Indexing - File Structure

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

- The spatial access method, holding an f-dimensional vector for each stored image (ARG). There is only one R-tree entry per image. In [10], every image produces multiple R-tree entries. We used R-trees solely because of availability; *any* spatial access method would do, like, e.g., X-tree [24] or SR-tree [25]. A faster access structure would only make our method run 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 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. 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 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).

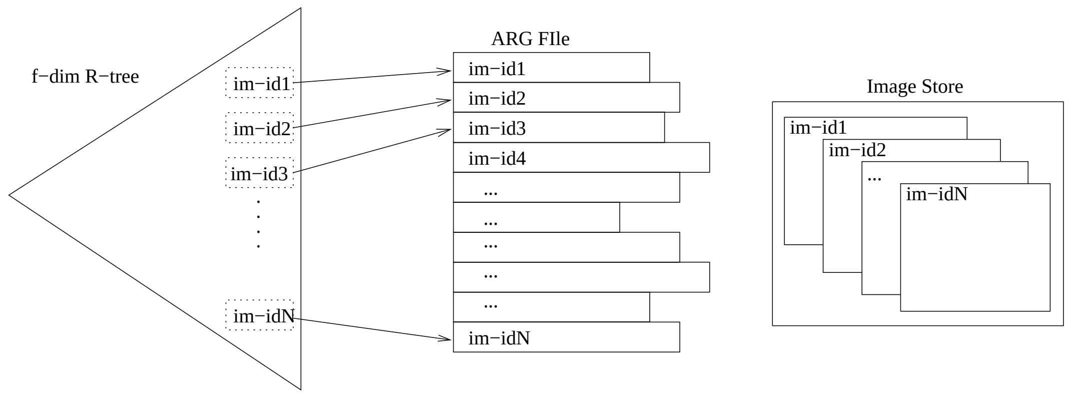

Figure 6: File structure.

Clean-up: For each of the above obtained images, retrieve its corresponding ARG from the ARG file and compute the actual (Eshera-Fu) 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.

5 Experiments

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 [10] 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 R-tree take 278.5 Kbytes for storage 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. 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. The times reported correspond to the elapsed time computed using the `time` system call of UNIX. 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 compared to sequential scanning. We show that our method produces almost always the same responses that the sequential scan method produces, but much faster.

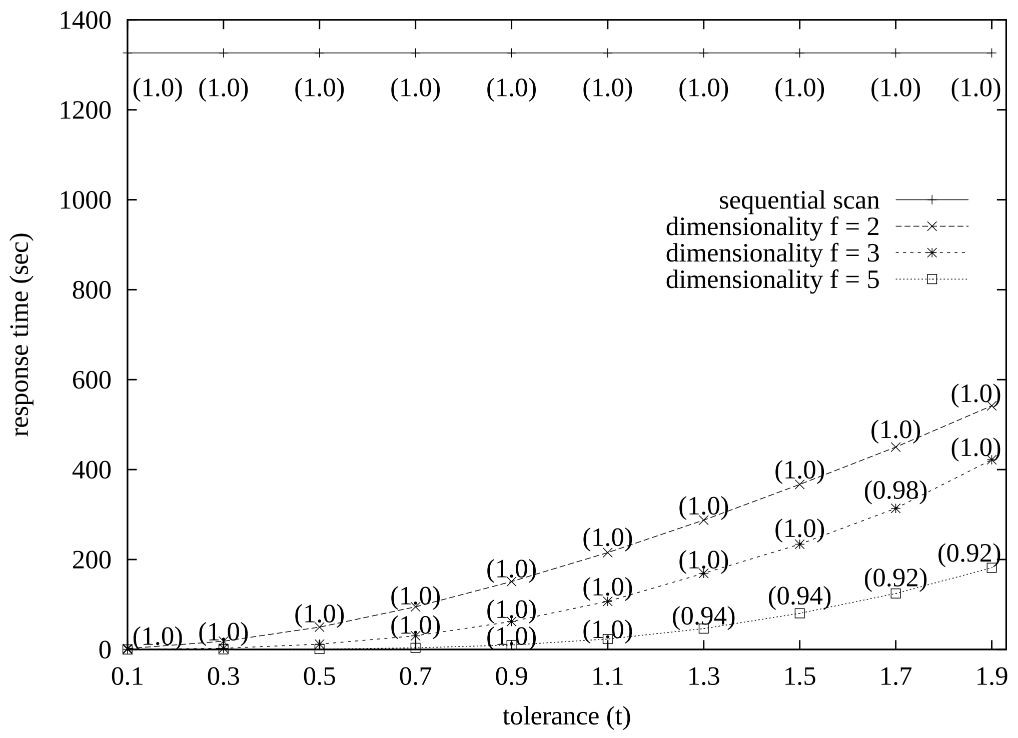

Figure 7: Average retrieval response time on the SYNTHETIC dataset, corresponding to indexed and sequential scan searching, as a function of the tolerance t, for dimensionality f=2, 3 and 5. The labels denote the average value of accuracy.

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. Sequential scan takes the same time, for any tolerance, as expected. For our proposed method, the effort increases with the tolerance. For $t = 2$, 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. 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.

5.2 False Dismissals

The labels in Figure 7 denote the average value of accuracy. Accuracy is always 1 (i.e., 100%) for sequential scanning. For the proposed method the accuracy 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%,

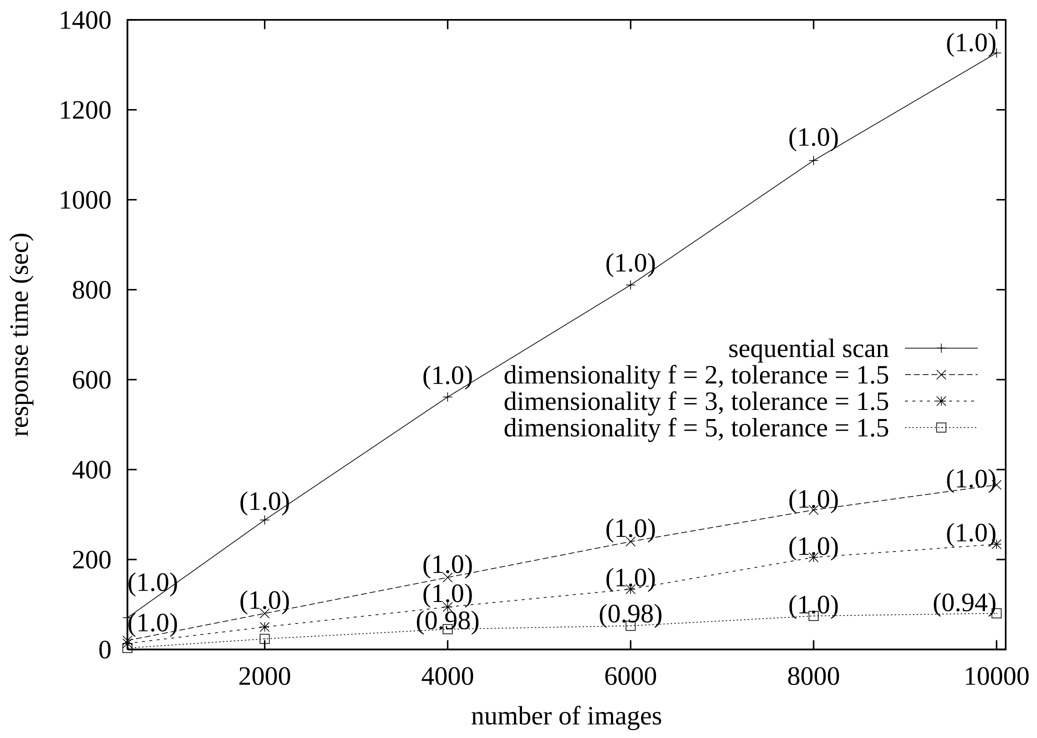

Figure 8: Average retrieval response time on the SYNTHETIC dataset, corresponding to indexed and sequential scan searching, as a function of the database size, for tolerance $t = 1.5$ and dimensionality $f = 2$, 3 and 5. The labels denote the average value of accuracy.

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 accuracy. In all cases, the accuracy 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. There are 5 clusters corresponding to images 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.

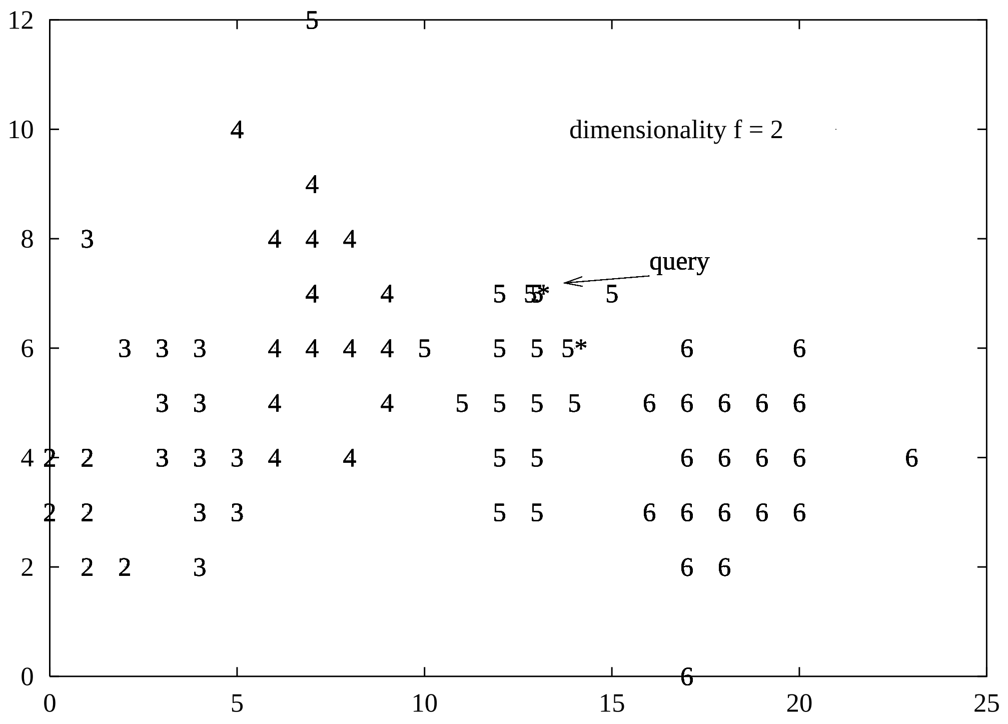

Figure 9: Diagram for 124 real MRI images (REAL dataset). There are 5 clusters corresponding to images with 2, 3, 4, 5 and 6 regions respectively. The labels denote number of regions in images. The arrow denotes the projection of the query of Figure 10. Its 3 nearest neighbors (also shown in Figure 10) are denoted by “$5*$”.

The diagram of Figure 9 can be found useful for browsing: 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. 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 on the 2-dimensional space. The query has 5 regions and 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). Two out of the three nearest neighbors share the same position on this diagram so that, only two “$5*$” are shown.

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.

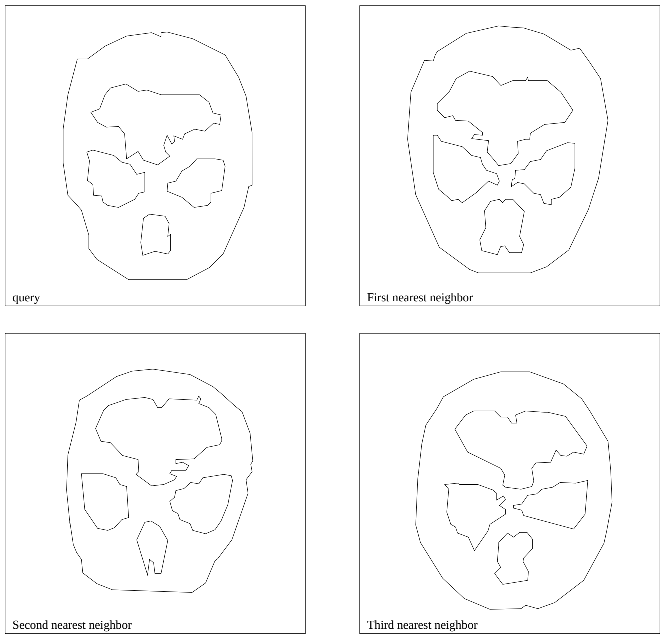

Figure 10: Example of a query image (top left) and of its 3 nearest neighbors.

- It introduced no false alarms and few, or no false dismissals.
- It allows for visualization, browsing and data-mining on image database contents.
- Completes and extends previous work by the authors [10].

With respect to future work, a very promising direction is the study of data-mining algorithms [26] 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 X-tree [24] or the SR-tree [25], which promise faster search times for smaller space overhead.

References

[1] A. Del Bimbo. *Visual Information Retrieval*. Morgan Kaufmann Publishers, Inc., 1999.

[2] A. W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-Based Image Retrieval at the End of the Early Years. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(11):1349–1380, Dec. 2000.

[3] S. Shekhar, S. Chawla an S. Ravada, A. Fetterer, X. Liu, and C.-T. Lu. Spatial Databases - Accomplishments and Research Needs. *IEEE Trans. on Knowledge and Data Engineering*, 11(1):45–55, Jan./Feb. 1999.

[4] A. Gupta and R. Jain. Visual Information Retrieval. *Comm. of the ACM*, 40(5):71–79, May 1997. (http://www.virage.com).

[5] M. Flickner et. al. Query By Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):23–32, Sept. 1995. (http://wwwqbic.almaden.ibm.com).

[6] A. Pentland, R. W. Picard, and A. Sclaroff. Photobook: Content Based Manipulation of Image Databases. *Int. Journal of Comp. Vision*, 18(3):233–254, 1996. (http://vismod.www.media.mit.edu/vismod/demos/photobook/index.html).

[7] B. T. Messmer. *Efficient Graph Matching Algorithms*. PhD thesis, Univ. of Bern, Switzerland, 1995. (http://iamwww.unibe.ch/˜fkiwww/projects/GraphMatch.html).

[8] E. G.M. Petrakis, C. Faloutsos, and King-Ip (David) Lin. Image Indexin Based on Spatial Similarity. Technical Report MUSIC-TR-01-99, Multimedia Systems Institure of Crete (MUSIC), April 1999. (http://www.ced.tuc.gr/˜petrakis).

[9] M. A. Eshera and K.-S. Fu. A Graph Distance Measure for Image Analysis. *IEEE Trans. on Systems, Man, and Cybernetics*, 14(3):353–363, 1984.

[10] E. G.M. Petrakis and C. Faloutsos. Similarity Searching in Medical Image Databases. *IEEE Trans. on Knowledge and Data Engineering*, 9(3):435–447, May/June 1997.

[11] 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://iamwww.unibe.ch/˜fkiwww/publications/index.html# technicalreports).

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

[13] S.-K. Chang, Q.-Y. Shi, and C.-W. Yan. Iconic Indexing by 2-D Strings. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(3):413–428, May 1987.

[14] J. R. Smith and S.-Fu Chang. VisualSEEk: A Fully Automated Content Based Image Query System. In *Proc. ACM Mult. Conf.*, Boston Ma., Nov. 1996. (http://disney.ctr.columbia.edu/VisualSEEk).

[15] E. G.M. Petrakis and S. C. Orphanoudakis. Methodology for the Representation, Indexing and Retrieval of Images by Content. *Image and Vision Computing*, 11(8):504–521, Oct. 1993. (http://www.ced.tuc.gr/˜petrakis).

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

[17] E. G.M. Petrakis. Design and Evaluation of Spatial Similarity Approaches for Image Retrieval. *Image and Vision Computing (to be published)*, 2000. (http://www.ced.tuc.gr/˜ petrakis).

[18] W. J. Christmas, J. Kittler, and M. Petrou. Structural Matching in Computer Vision Using Probabilistic Relaxation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8):749–764, August 1995.

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

[20] E. A. El-Kwae and M. Kabuka. A Robust Framework for Content-Based Retrieval by Spatial Similarity in Image Databases. *ACM Trans. on Information Systems*, 17(2):174–198, April 1999.

[21] E. A. El-Kwae and M. Kabuka. Efficient Content-Based Indexing of Large Image Databases. *ACM Trans. on Information Systems*, 18(2):171–210, April 2000.

[22] D. Papadias, N. Mamoulis, and V. Delis. Algorithms for Querying by Spatial Structure. In *Proc. of 24th VLDB Conf.*, pages 546–557, NY, USA, 1998.

[23] C. Faloutsos and K.-I. Lin. FastMap: A Fast Algorithm for Indexing, Data Mining and Visualization of Traditional and Multimedia Datasets. In *Proc. of ACM SIGMO*, pages 163–174, San Jose, California, June 1995.

[24] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree : An Index Structure for High-Dimensional Data. In *Proc. of 22nd VLDB Conference*, pages 28–39, 1996.

[25] N. Katayama and S. Satoh. The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. In *Proc. of ACM SIGMOD*, pages 369–380, 1997.

[26] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proc. of VLDB Conf.*, pages 487–499, Santiago, Chile, September 1994.

Contents

1 Introduction **2**

2 Related Work **3**

3 Attributed Relational Graphs (ARGs) **4**

3.1 ARG Editing Distance 5

4 Proposed Method **7**

4.1 Image Segmentation 7

4.2 ARG Representation 8

4.3 Mapping ARGs to f-dimensional Points - FastMap 8

4.4 Indexing - File Structure 9

4.5 Search Strategy 9

5 Experiments **10**

5.1 Response Time 11

5.2 False Dismissals 11

5.3 Scale-up 12

5.4 Visualization 12

6 Conclusions **13**