Fast Retrieval by Spatial Structure in Image Databases

Euripides G.M. Petrakis

Department of Electronic and Computer Engineering

Technical University of Crete

Chania, Crete, Greece

petrakis@ced.tuc.gr

February 26, 2002

Abstract

The state-of-the-art approach for speeding-up the time responses in databases is using *Spatial Access Methods* (SAMs) like e.g., R-trees. However, these methods do not treat image content directly (e.g., objects are approximated by their minimum bounding rectangles), nor can they handle images with multiple regions. The proposed approach extends the existing framework of indexing using SAMs to treat image content in conjunction with two well known image matching methods, namely the *editing distance* on Attributed Relational Graphs (ARGs) and the *Hungarian* method for graph matching. It provides index support for the two most common types of similarity queries, referred to as *range* and *nearest neighbor* queries and has many desirable properties. For instance, it handles even complex queries specifying multiple objects (such as queries by image example), it returns exactly the same answers with the sequential scan methods (without indexing) and works with any SAM (e.g., R-tress) and with any image distance function provided that it satisfies the so called *Lower Bounding Principle*.

Index terms: image database, image indexing, range query, nearest neighbor query, attributed relational graph, spatial access method.

1 Introduction

The management of large volumes of digital images in many application fields (e.g., medicine [1], criminal investigation [2], trademark/copyright detection [3] etc.) has generated additional interest in methods and tools for real time archiving and retrieval of images by content. Consequently, content-based image retrieval has become the object of intensive research activities over the past few years [4, 5]. Several approaches to the problem of content-based image management have been proposed and some have been implemented on research prototypes and commercial systems [6, 7, 8, 9].

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

Retrievals by image content is not an exact process (images are rarely identical). Therefore, database search criteria must be approximate so that all images up to a prespecified degree of similarity are retrieved. Approximate database search is referred to in the literature as "similarity search". There are also two types of similarity queries known as *range* queries (i.e., find all images within a prespecified distance from the query) and *nearest neighbor* (NN) queries (i.e., retrieve the k best matches). The design of appropriate image distance functions is a key issue here. An image distance function must take advantage of the properties of its underlying image representation and must return images satisfying the query selection criteria. A method is successful if it retrieves images that the users expect to see in the answers, with as few errors as possible.

The obvious method to search an IDB is sequential scanning: The query is matched with all stored images (i.e., in fact, the representation of the query is matched with all representations stored in the IDB). Retrievals may become extremely slow, especially when database search involves time consuming image matching operations. To deal with slow retrieval response times and high complexity matching, an IDB system must utilize indexing methods that are faster than sequential scanning methods. To be practical, these methods must work in conjunction with range and NN queries. The processing of these two types of queries using Spatial Access Methods (SAMs) like

e.g., R-trees [10] requires substantially different algorithms than sequential scanning.

Summarizing, there are two general goals common to all IDB systems:

Effectiveness: An IDB system must treat images and queries of arbitrary content complexity (e.g., queries specifying multiple objects and their relationships) and must be *accurate* that is, it must retrieve similar images (i.e., images that the users expect in the answers) with as few errors as possible.

Efficiency: An IDB retrieval method must be *fast* (i.e., faster than sequential search methods) for both, range and NN queries.

The first issue is treated in [11]. The proposed work emphasizes on the speed of search and on image indexing by content. Attention is focused on images and queries specifying (probably) more than one regions (or objects). The textbook approach to capture this information is "Attributed Relational Graphs" (ARGs) [12]. ARGs are the most general representations of image content and among other existing retrieval methods (e.g., [13, 14, 15]) are the most accurate, achieving higher precision and recall than any other method [11]. In this work, ARGs are adopted as the underlying image content representation method in conjunction with the *editing distance* [16] and the *Hungarian method* [17] for image matching.

Focusing on the speed of search using ARGs, an approach is proposed for providing index support to range and NN queries on ARGs. Each database image (or query) is represented by multiple points (feature vectors) in a multidimensional space which is indexed using a SAM. Traditionally, SAMs assume that each data entity (or query) is represented by exactly one multidimensional point. Therefore, they cannot treat directly images with more than one objects ore regions. Two algorithms are proposed for handling range and NN queries respectively. Both algorithms guarantee no "false dismissals" and no "false drops" that is, they return exactly the same answers with the sequential method, only much faster. The proposed approach works with any image type and with any image distance function provided that it satisfies the "Lower Bounding Principle" (i.e., the distance in the feature space is lower than the actual image distance). Finally, the method works with any SAM (like R-trees [10] or SR-trees [18]). The efficiency of the method is demonstrated using R-trees. In fact, a faster SAM would only make this approach work even faster! The performance of the proposed approach in retrieving images by content is evaluated using the dataset of [1] and compared with the performance of sequential scanning using the ARG editing distance or the Hun-

garian method. The results demonstrate a very significant improvement in retrieval performance: Indexed search is many times faster than sequential scanning even for large answer sets (with more than 10 images).

The rest of this paper is organized as follows: A review of the related work is presented in Section 2. Issues related to image representation and image matching using ARGs are discussed in Section 3. The proposed indexing method is presented in Section 4. Experimental results are presented and discussed in Section 5 followed by conclusions in Section 6.

2 Related Work

Issues related to IDB system design and implementation have been raised previously by many investigators. Recent surveys have been published in [4, 5]. In the following, related issues are discusses separately for ease of presentation.

2.1 IDB Methods and Systems

In the Virage system [6], image content is given primarily in terms of properties of color and texture. The QBIC system of IBM [7] utilizes a retrieval method for queries on color, texture and shape. The Photobook [8], the system developed at the MIT Media Lab, supports queries by image content in conjunction with text queries. VisualSEEk [9] is a visual query system for searching for color photographic images. It is based on three separate search engines, SaFe for retrieving images by spatial content, CBVQ for retrieving images based on color and texture properties and WebSEEk for searching in the WWW. Chu et.al. [19] describes a system for retrieving medical images by spatial and temporal content. A desirable feature common to many systems is the adaptive behavior to retrievals through user relevance feedback and iterative query refinement (e.g., [20]).

Additional work on IDB search and retrieval includes the work by Ratha et.al. for fingerprints [2]. Das et.al. [21] and Huang et.al. [22] suggest using the color image attributes as measurements of image content in image databases. Mojsilovic et.al. [23] focuses on the perception of color by humans and proposes a method that attempts to simulate human behavior in matching and retrieving color patterns. The authors detected five visual categories that people use in judgment

of similarity and they proposed a set of rules governing the use of these categories.

2.2 Retrieval by Shape Content

Regarding IDB retrieval by shape content, [24] reports experiments with traditional shape representation and matching methods (e.g., Fourier, moments) on 500 trademark images. More recently, the effectiveness of such shape methods in conjunction with color features is investigated in [3] using 1,100 trademark images. SQUID [25] supports retrievals on a dataset of 1,100 marine life species. [26] demonstrates the superiority of a multiscale dynamic programming matching method over Fourier and moment-based methods on 980 static hand gesture shapes and on the dataset of SQUID.

Korn et.al. [27] proposes a method for retrieving similar shapes from an IDB based on mathematical morphology. The method supports the indexing of shapes by R-trees by taking the components of the "Pattern Spectrum" as features of a multidimensional vector. Bimbo et.al. [20] presents a system for the interactive retrieval of shapes. A set of global shape features are used to narrow down the search into a small number of candidate shapes which are then matched with the query using a sampled point representation (signature) of the shape contour. Additional work on shape retrieval include PicToSeek [28] a system which combines shape and color features for shape retrieval and the work by Sclaroff [29]. The novelty of the last method is that database shapes are indexed based on their distance from a few shape prototypes corresponding to the main shape categories.

2.3 Retrieval by Spatial Relationships

Focusing mainly on color, texture and shape, the work referred to above does not show how to treat multiple objects or regions in image queries, nor their interrelationships. In this case, for two images to be similar, not only the shape, color and texture properties of individual image regions must be similar, but they must have the same arrangement (spatial relationships) in the two images. The textbook approaches to treat this information in IDB systems are "2D strings" and "Attributed Relational Graphs" (ARGs) [12].

2D strings [13] and their variants [30] are examples of work focusing mainly on spatial relationships and provide low complexity (i.e., polynomial) matching in IDBs. 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) [14]. 2D strings are an adequate representation of spatial image properties in the case of images consisting of non-overlapping objects (i.e. their minimum enclosing rectangles do not intersect) having simple shapes. 2D C strings [31] handle overlapping objects with complex shapes but they are not as simple and compact as the original 2D strings nor can they can be produced very efficiently. Besides, the matching of 2D C strings has exponential time complexity.

SaFe [32] combines region with spatial queries utilizing 2D strings along with color and texture features of selected regions. The overall match cost is computed by adding the weighted distances computed on all properties and spatial relationships. SaFe has been evaluated using two sets of 500 synthetic and 3,100 real color images respectively. It has been shown to perform better than systems using color features alone.

In ARGs, individual objects or regions are represented by graph nodes and their relationships are represented by arcs between such nodes. Both nodes and arcs may be labeled by attributes. The method by Petrakis and Faloutsos [1] utilizes ARGs and achieves fast retrievals by content by exploiting the assumption that each image always has a fixed number of “expected” objects and by indexing ARGs with R-trees.

The method by Gudivada and Raghavan [15] 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 [33] 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 their interrelationships into account [16].

A comparison of the proposed approach on image indexing with recent relevant work in ARG matching and image indexing by content is also discussed in Section 4.4.

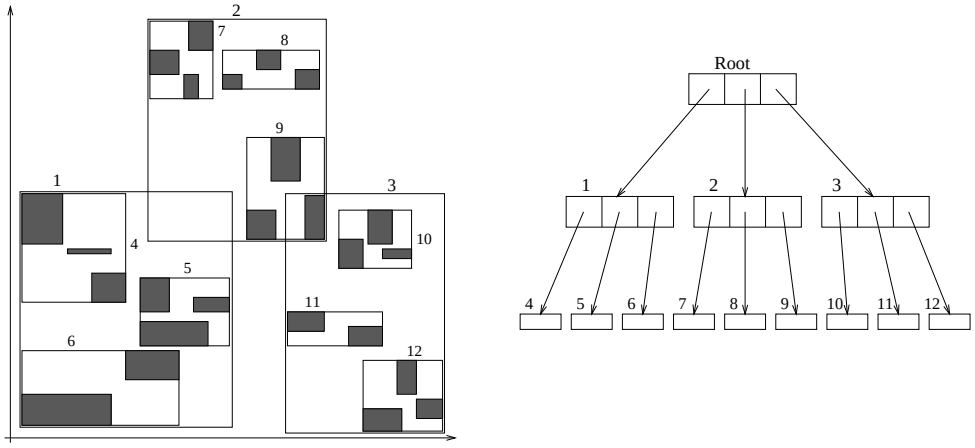

Figure 1: *Data (dark rectangles) organized in an R-tree.*

2.4 Spatial Access Methods (SAMs)

The so called "Spatial Access Methods" (SAMs) are file structures to manage large collections of f-dimensional points (or rectangles or other geometric objects) in the main memory or 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. Spatial access methods can also treat nearest neighbor [34] and "spatial-join" queries [35]. Several spatial access methods have been proposed [36]. One of the most characteristic approaches in the last class is the R-tree [10]. Extensions, variations and improvements to the original R-tree structure include (among others) the R*-tree [37] and recently, the the X-tree [38] and the SR-tree [18].

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 guarantees good space utilization and remains balanced. Figure 1 illustrates data rectangles (in black) organized in an R-tree (left); the file structure for the same R-tree is also shown (right); the nodes correspond to disk pages. The example is due to [27].

In the proposed work, image content is represented by a list of f features. These features are

mapped into points in an f-dimensional space and the data objects are points instead of rectangles which can also by handled by a SAM. Then a range query requires all the points that are inside the region of interest and a NN query requires the k data points closest to the point of the query.

3 Image Representation

It is assumed that each image has been segmented (automatically or manually) to more than one objects or regions. Each object or region is represented by its surrounding contour. Figure 2 shows an example of an original grey-level image (left) and its corresponding segmented form (right). These segmented forms are used to compute a variety of image features specific to a particular image representation. Automatic image segmentation is a nontrivial problem and it is outside the scope of this paper. In this work we used the images of [1] which are already available in the desired segmented form.

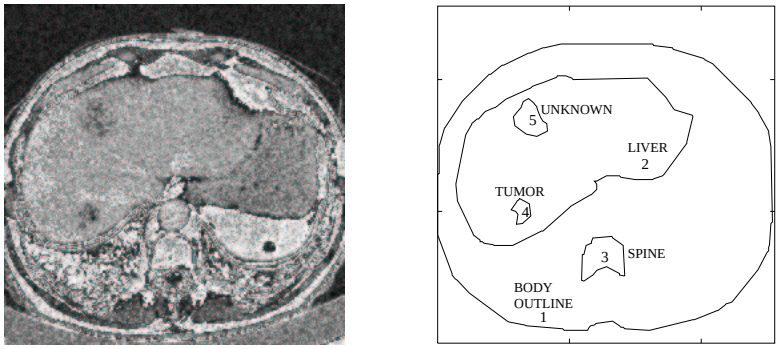

Figure 2: *Example of an original grey-level image (left) and its segmented form (right).*

3.1 Attributed Relational Graphs (ARGs)

Figure 3 illustrates the ARG computed for the image of Figure 2. Nodes correspond to regions and arcs correspond to relationships between regions. The exact form of the ARGs is determined by a domain expert. A representation that has been used successfully [1, 39] is as follows: Arcs are directed from the outer to the contained region but their direction also depends on object labels. In Figure 3, the arcs are always directed from body outline to the remaining objects, from liver to spine and from the most common objects (i.e., body outline, liver and spine in this application) to the remaining objects. Both nodes and arcs are labeled by attributes corresponding to properties

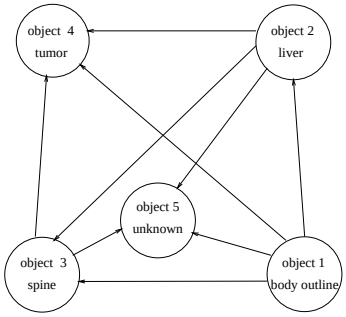

Figure 3: *Example ARG.*

(features) of objects and relationships respectively. A set of features that has been also used in [14, 1, 39] is the following:

Individual regions are represented by *size (s)*, computed as the size of the area it occupies, *perimeter (p)* computed as the length of its bounding contour, *roundness (r)*, computed as the ratio of the smallest to the largest second moment and *orientation (o)*, defined to be 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 the following set of features: *relative distance (rd)*, computed as the minimum distance between their surrounding contours, *relative orientation (ro)* defined as the angle with the horizontal direction of the line connecting the centers of mass of their regions and *relative position (rp)* taking values $(-1, 0, 1)$ corresponding to regions which are the first one inside the other (-1), outside each other (0) or, the second inside the first one (1) respectively.

To avoid discontinuities in the measurement of angles (i.e., orientations of 1 degree should have measurements similar to those of orientations of 359 degrees), angles are represented by both their *sin* and *cos*. All attributes are normalized in the range $[0, 2]$ (i.e., maximum feature distance). Notice that, a range $[0, 2]$ places no additional constraints compared with the typical and most commonly used range $[0, 1]$. Distances and areas are divided by the half of the perimeter and area of the largest region (i.e., body outline) respectively. Roundness takes values in the range $[0, 1]$ and is multiplied by 2. This normalization results into features which are scale invariant. These features are also translation invariant since only relationships between objects are used to characterize object positions. To achieve rotation invariance, all images are registered to a standard

orientation (e.g., the axis of elongation of the outline object is made horizontal). Individual objects or regions are represented by 5-dimensional vectors of the form $(s, p, r, 1 + cos(o), 1 + sin(o))$ while, relative orientations are represented by 4-dimensional vectors of the form $(rd, 1 + rp, 1 + cos(ro), 1 + sin(ro))$.

An ARG representation can treat any set of features that a domain expert may provide such as, grey-level and texture values, moments or Fourier coefficients etc., as node labels, relative size, relative orientation, amount of overlapping or adjacency etc., as edge labels. However, the emphasis of this work is not on choosing a good set of features, but in accelerating the search for a given set of attributes.

3.2 ARG Editing Distance

Matching a query and a stored ARG is treated as an *"error-correcting subgraph isomorphism"* problem [16]: Given their graphs Q (query) and I (model or reference ARG) there is always a sequence of "edit" operations (or corrections) $S(I) = \delta_l(\delta_{l-1}(\ldots\delta_2(\delta_1)\ldots))$ that transform I to a subgraph of it which is isomorphic to Q (i.e., there may exist extra nodes or edges in I but not in Q). These edit operations take the form of node or edge deletions on I (equivalently insertions on Q) and node or edge substitutions. There are infinite sequences of such edit operations and one would like to choose the *best* one. This can be achieved by assigning costs to edit operations, combining the costs of a sequence $S(I)$ in a meaningful way and by taking the sequence yielding the minimum cost. More formally, the distance $D(Q, I)$ between a query Q and a model graph I is defined as

$$D(Q, I) = \min_{S(I)} \{\Phi(S(I))\} = \min_{S(I)} \{\Phi(\phi(\delta_l), \phi(\delta_{l-1}), \ldots \phi(\delta_1))\}, \quad (1)$$

where Φ is a function that combines the costs of all the l edit operations δ in $S(I)$ and ϕ is a function that computes the cost of an edit operation δ_i, $1 \leq i \leq l$. The definition of cost functions Φ and ϕ depends on the application, the edit operations allowed and on the labels used. Traditionally, Φ is defined as a summation of ϕ costs:

$$\Phi_{sum} = \Phi(S(I)) = \sum_{i=1}^{l} \phi(\delta_i). \quad (2)$$

Alternatively, Φ can be defined as the maximum of the above ϕ costs:

$$\Phi_{max} = \Phi(S(I)) = \max_{i=1}^{l} \{\phi(\delta_i)\}. \quad (3)$$

If nodes and edges are represented by attribute vectors (as it happens in this work), ϕ can be defined as a vector distance and is computed using an L_p metric. If $v = (z_1, z_2, \ldots z_w)$ and $v' = (z'_1, z'_2, \ldots z'_w)$ are two such vectors then

$$\phi_p = \phi(\delta) = \left[\frac{1}{w} \sum_{i=1}^{w} |z_i - z'_i|^p \right]^{1/p}, \tag{4}$$

where p is the order of the metric. For $p = 1$ and $p = 2$ the Manhattan ϕ_1 (city-block) and the Euclidean ϕ_2 distances respectively are obtained. If $p \to \infty$, the Chebeychev distance ϕ_∞ is obtained:

$$\phi_\infty = \phi(\delta) = \max_{1 \le i \le w} \{|z_i - z'_i|\}. \tag{5}$$

Notice that $\phi_1 \le \phi_\infty$, $\phi_2 \le \phi_\infty$ and $\phi \le \Phi$, for any definition of ϕ and Φ respectively. There may exist extra nodes or edges in I (in the stored ARG) but not in the query ARG Q. Extra nodes or edges in I are ignored (i.e., their cost is 0) while, extra nodes or edges in Q are not allowed (i.e., their cost is ∞).

The computation of the distance between two ARGs involves not only finding a sequence of error transformations, but also finding the one that yields the minimum total cost. This can be formulated as a tree search problem which can be solved by an $A^\star$ algorithm [16]. In the following, the ARG at the left of Figure 4 (query ARG) is matched with the ARG at the right (model ARG). All nodes and all edges are labeled by their attribute vectors consisting of two attributes taking values in the range [0,2]. The algorithm creates the state-space tree of Figure 5. Each state (tree node) corresponds to a matching of a pair of subgraphs from the two input ARGs. A transition from a state to another corresponds to the embedding of a pair of unmatched nodes (one from each ARG) into the already matched subgraphs. Each state in Figure 5 is labeled by a pair of nodes (in parenthesis) and by the cost of matching these nodes (in square brackets). Transitions are labeled by the costs of matching the relationships of the added nodes with all the nodes currently on the path.

The state-space tree expands until a complete match is found. A complete match is one that has consumed all query nodes (but not necessarily all model nodes). The minimum total cost found at any time can be used as an upper bound to prune the expansion of non-promising paths yielding partial cost greater than the minimum total cost found so far.

The edit operations for each embedding are recorded and their costs are accumulated in a meaningful way. In Figure 5 node and edge matching costs in Figure 5 are computed according

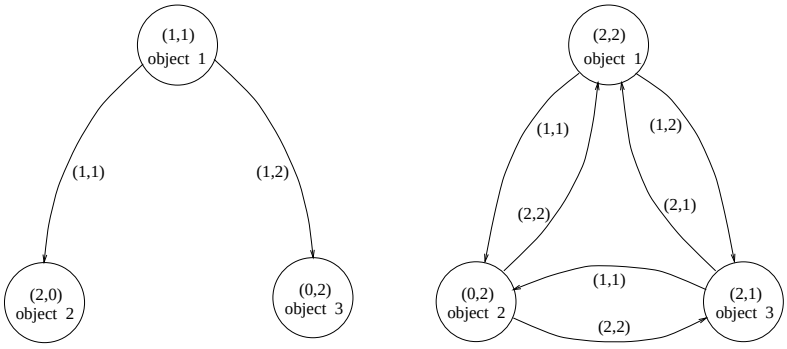

Figure 4: *Example query ARG (left) and reference ARG (right).*

to Equation 5. The editing distance (node and edge costs along the same path) are computed according to Equation 2. In this example, query node 1 is associated with model node 1, query node 2 is associated with model node 3 and query node 3 is associated with model node 2. The cost of this matching is 4 (best cost). If the maximum of all the node and edge costs along a path is taken, the best cost is 1 (all other paths have cost 2).

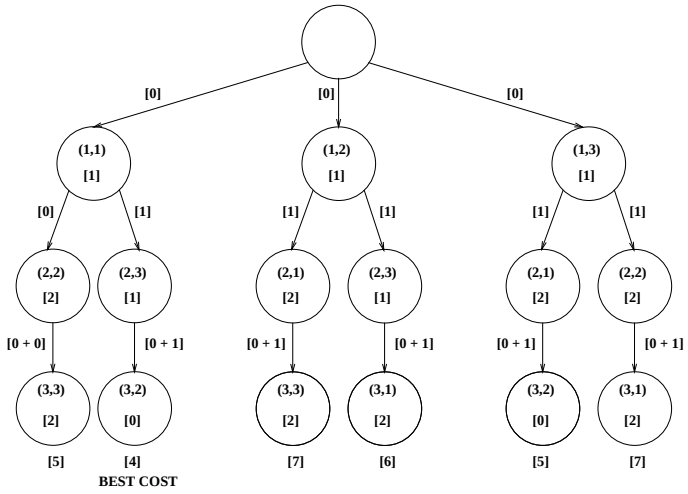

Figure 5: *Matching tree.*

The method for ARG matching referred to above finds the optimal solution but it has exponential time (and space) complexity in the worst case (i.e., the matching tree can grow exponentially for large graphs). Polynomial time algorithms do exist but, either they are approximate [40] or require exponential space [41].

3.3 Hungarian Method

Matching between two ARGs can also be formulated as an *assignment* problem, that is a problem of finding the best association between the nodes (objects) of the query and the model ARG (the relationships are ignored). The cost of this association, is computed based on the costs (weights) $C(i,j)$ of associating node i in Q with node j in I, $i,j \leq n$, where n in the number of nodes in the two graphs. $C(i,j)$ is computed as the L_p distance between their corresponding feature vectors using Equation 4 or Equation 5.

Let $\mathcal{F}()$ be a mapping from nodes in Q to nodes in I. The cost of this mapping is defined as:

$$D_{\mathcal{F}}(Q,I) = \sum_{i=1}^{n} C(i,\mathcal{F}(i)). \tag{6}$$

The distance between the two ARGs is defined as the minimum distance computed over all possible mappings $\mathcal{F}()$ and corresponds to the best association of nodes in Q with nodes in I:

$$D(Q,I) = \min_{\mathcal{F}} \{D_{\mathcal{F}}(Q,I)\}. \tag{7}$$

Figure 6 illustrates a bipartite graph which is constructed from a query Q and a model ARG I by associating each object in Q with all objects in I. The labels on the arcs connecting objects in Q with objects in I correspond to the costs $C(i,j)$. The best mapping $\mathcal{F}()$ is computed using the *Hungarian* method in $O(n^4)$ time [17]. Solid lines in Figure 6 correspond to the best mapping. The cost of this mapping is 10. This is the cost of associating object 1 in Q with object 2 in I, plus the cost of associating object 2 with object 1, plus the cost of associating object 3 with object 4 and, finally, object 4 with object 3.

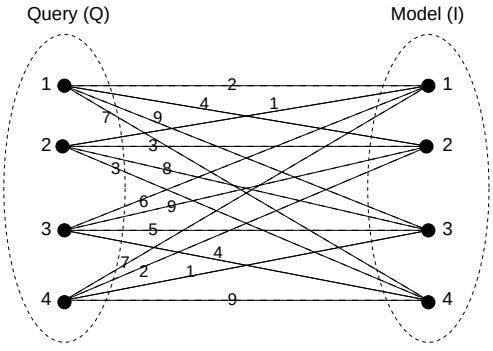

Figure 6: *Image matching as an assignment problem.*

The above formulation assumes that the two ARGs have the same number of nodes. However, if Q contains fewer nodes that I (as it happens in this work), any missing objects are added in Q,

assuming that the costs $C(i, j)$ incident upon them are all ∞. If Q contains more objects than I, then, $D(Q, I)$ is ∞ by definition (i.e., extra objects are not allowed in Q).

4 Image Indexing by Content

To speed-up retrieval response times the stored ARGs are indexed. The main idea is to extract f features from each image (or query) and map these features to points in an f-dimensional space. In this work R-trees are used as the underlying spatial access structure. R-trees are used solely because of availability; *any* SAM (like the SR-tree [18]) would do (a faster method would make the method even faster!).

A SAM requires that the number of dimensions f is known in advance and it is fixed. In ARGs, individual objects (or relationships) are represented by a fixed number of features. However, the number of objects is not the same in all images. Therefore, ARGs, as a whole, cannot be represented by fixed dimensionality vectors and cannot be indexed directly using SAMs. To deal with this problem, each image (or query) is decomposed into "subimages" (or "subqueries").

Definition 1 *A subimage of a given image contains a fixed number of objects.*

In this work, the subimages and the subqueries contain exactly one object. This is a general setting: The proposed approach is independent of the definition of subimages and subqueries and of the definition of the distances Φ and ϕ (Section 3.2). For example, the image matching method of [42] 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 (relationships) emanating from it and the nodes (objects) on which these arcs terminate (terminal or leaf nodes). For this method, each subimage corresponds to a BARG and represents both objects and relationships. Similarly, in [1] each subimage contains a fixed number of known objects plus one or two unknown (together with their relationships). For these methods, the distances Φ and ϕ have to be defined on BARGs or on subimages with more than one objects respectively.

Each subimage (or subquery) is represented by its f-dimensionality feature vector ($f = 5$ in this work) and corresponds to a point in an f-dimensional feature space. Equivalently, each image

(or query) is represented by n points in the same f-dimensional space where, n is the number of subimages (objects in this setting) in the image (or query).

Figure 7 illustrates the mapping of three images (denoted as I,J, L) and one query (denoted as Q) in a 2-dimensional space ($f = 2$). Image I consists of three regions (equivalently subimages) and it is mapped to three 2-dimensional points (the I_1, I_2, I_3). J has one object (subimage) and it is mapped to one point (the J_1). Finally, image L consists of two objects (subimages) and it is mapped to two 2-dimensional points (the L_1 and L_2). Similarly, query Q specifies two objects (equivalently, two subqueries) and it is mapped to two 2-dimensional points (the Q_1 and Q_2). The black dots denote the positions of the subimages and of the subqueries in the above 2-dimensional space. The rectangles centered around Q_1 and Q_2 denote range queries with distance tolerance t on each dimension. The processing of range and NN queries in an IDB can be treated through the processing of subqueries in the f-dimensional space. In the above example, a range query with distance tolerance t on Q_1 would retrieve the subimages L_1 and L_2. Similarly, a range query on Q_2 would retrieve only one subimage, the I_1. The retrieved subimages are used to hypothesize that images J and L (but not I since no subquery retrieved a subimage derived from I) are candidate images for matching the whole query Q. The details of the algorithms are discussed in Section 4.2 and in Section 4.3. An ϕ_2 query would specify a circular query of radius t enclosed by the above rectangles.

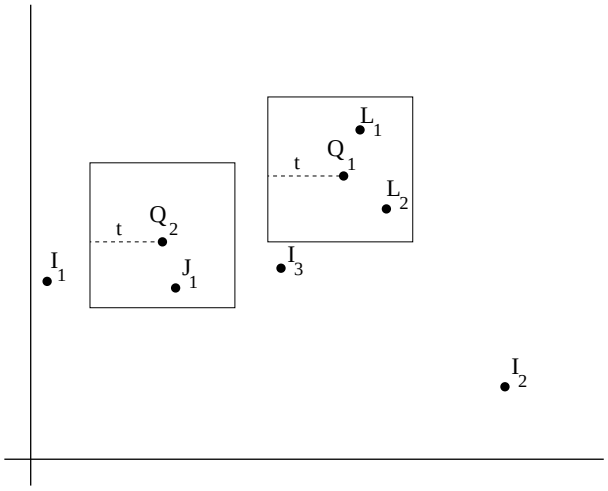

Figure 7: *Mapping images and queries to points in a 2-dimensional space.*

The file structure on the disk consists of two parts:

- The SAM (i.e., an R-tree) holding the f-dimensional vectors (subimages) derived from each

stored image and

- The "ARG file", which is a sequential file holding all the ARGs. There are pointers from the subimages in the R-tree pointing to the ARGs (in the ARG file) from which they have been derived.

4.1 Basic Definitions

A query Q is decomposed into n subqueries $Q_1, Q_2, \ldots Qn$; this is denoted as $Q = (Q_1, Q_2, \ldots, Q_n)$. Similarly, each stored image (model) I is decomposed into m subimages $I_1, I_2, \ldots, I_m$; this is denoted as $I = (I_1, I_2, \ldots, I_m)$. In general $n \neq m$.

Two types of distance are defined:

Feature distance $D_f(Q_i, I_j)$: It is the distance between the Q_i and I_j derived from Q and I respectively. D_f is used to search the SAM and can be defined as an ϕ_1 (Manhatan), ϕ_2 (Euclidean) or an ϕ_∞ (Chebeychev) vector distance. In the example of Figure 7, $D_f = \phi_\infty$. Henceforth, for convenience, missing subscripts from Q_i (or I_j) denote feature distance between any subquery (or subimage) derived from Q (or I).

Actual distance $D(Q, I)$: It is the actual distance between Q and I. In this work, $D(Q, I)$ can be either an ARG editing distance (defined as an Φ_{sum} or an Φ_{max} operation on ϕ costs) or a Hungarian distance (defined as a summation of ϕ costs). Henceforth, for convenience, $D(Q_i, I_j)$ denotes the actual distance between the Q and I from which the Q_i and I_j have been derived.

There are also two types of queries:

Range Queries: Given a query Q, retrieve all images I satisfying $D(Q, I) \leq t$ (t is user defined).

Nearest Neighbor (NN) queries: Given a query Q, retrieve its k most similar images with respect to distance D (k is also user defined).

The problem with range queries is that a good value of t cannot be known in advance. The answer sets can be empty if t is small or very large for large t. As shown in the following, both these types of queries can be answered through a SAM. Notice that, existing search methods on

SAMs (e.g., [10, 34, 43]) return database entities within distance t from the query (range queries) or its k NNs (NN queries) with regard to distance D_f not to D. However, in image similarity retrieval, image distances must be computed with regard to D. In addition, they assume that all images and the queries are represented by exactly one point in a multidimensional space. Therefore, such methods cannot treat image content nor can they be applied directly for indexing and retrieving images with multiple regions.

Definition 2 (Lower Bounding Principle:) *The distance between a query $Q = (Q_1, Q_2, \ldots, Q_n)$ and a stored (model) image $I = (I_1, I_2, \ldots, I_m)$ satisfies the lower bounding principle if $D_f(Q_i, I_j) \leq D(Q, I)\ \forall i, j$.*

Lemma 1 *The definitions of image distance of Section 3.2 and of Section 3.3 satisfy the lower bounding principle.*

Proof 1 *The ARG editing distance D is defined as a function of D_f terms (i.e., summation or maximum). Similarly, the Hungarian distance is defined as a summation of D_f terms. Obviously, $D_f \leq D$ for both definitions of image distance.* □

It is a responsibility of the system designer to ensure that an image distance satisfies the Lower Bounding Principle. Notice for example that, a non-normalized ϕ_p (Equation 4) in conjunction with Φ_{max} (Equation 3) does not satisfy this principle. Notice also that, the algorithms for range and NN search below are not restricted to subimages and subqueries with only one object. The only requirement is that the distance D_f satisfies the Lower Bounding Principle.

Table 1 summarizes the notation used in this work.

4.2 Range Queries

The basic idea in processing range queries in an IDB is to search the index (equivalently, the f-dimensional space) for each Q_i using D_f and to combine their results. Figure 8 illustrates an algorithm to retrieve all images I within distance t from Q.

Lemma 2 *The algorithm of Figure 8 guarantees no false dismissals.*

notation	meaning
t	distance tolerance
k	number of nearest neighbors (NNs)
$Q = (Q_1, Q_2, \ldots Q_n)$	query image
$I = (I_1, I_2, \ldots I_m)$	model (stored) image
n	number of subqueries
m	number of subimages
f	number of features
ϕ	cost of an edit operation
Φ	cost of a sequence $S(I)$ of edit operations
$D(Q, I)$	actual distance between Q and I
$D_f(Q_i, I_j)$	feature (vector) distance between Q_i and I_j

Table 1: *Summary of notation used in this work.*

Proof 2 *Suppose that the algorithm missed image I although it satisfies $D(Q, I) \leq t$ (assumption). Let $I = (I_1, I_2, \ldots I_m)$. This is possible only when, at least one Q_i in step 2 missed I that is, $D_f(Q_i, I_j) > t, \forall j = 1, 2, \ldots m$. Due to the lower bounding principle $D_f(Q_i, I_j) \leq D(Q, I)$. By combining the last two inequalities $D(Q, I) > t$ which contradicts the assumption.* $\square$.

The lower bounding principle ensures that there are no false dismissals. However, this does not guarantee no false positives (false drops) that is, there may exist images I in set A (step 3) satisfying $D(Q, I) > t$. To clean-up the false drops, a post-processing step is required. This is implemented by step 4.

4.2.1 Discussion

Notice that not all subqueries need to be applied at step 2. At one extreme all subqueries are applied. The other extreme is to apply only one subquery. Both strategies introduce false drops. The first strategy, attempts to minimize the false drops but, involves excessive R-tree search and set intersections. The second strategy avoids R-tree search as much as possible, but it employees an expensive clean-up stage. Both strategies guarantee no false dismissals. A compromise is to

Input: Query Q, Distances D, D_f, Tolerance t.

Output: All images I satisfying $D(Q, I) \leq t$.

1. Decompose Q into $Q_1, Q_2, \ldots Q_n$ subqueries.
2. For each Q_i apply a range query $D_f(Q_i, I) \leq t$ on the SAM. Each Q_i retrieves a set A_i of image identifiers I satisfying the range query.
3. Compute the answer set $A = \cap_{i=1}^{n} A_i$ of common image identifiers I.
4. For each I in A compute $D(Q, I)$ and output all images I satisfying $D(Q, I) \leq t$ (clean-up).

Figure 8: *Algorithm to retrieve all images within distance t from a query using a SAM.*

apply more than one but less than n subqueries. The number of subqueries that achieve faster retrieval depends on the dataset and can be specified by experimentation. In this work, the number of subqueries which resulted in faster retrievals is 2.

The above search scheme can also treat cases where the query specifies only a few of the f attributes (e.g., we don't care for some of them). The unspecified features are excluded from D_f. Notice that the lower bounding principle still holds in this case (i.e, the new D_f is derived from the original by omitting some positive terms). The R-tree index can still handle these queries: The range of values along the unspecified axes stretches from $-\infty$ to $+\infty$. This method can also handle the case where the user considers some of the properties more important than others (i.e., by assigning higher weights to these properties). If weights are used with Equation 4, 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.

The index implements a preliminary search step which is used to limit the search for qualifying answers to a small number of promising images. The lower the number of promising images the faster the method is. The method seems to be better suited for images with a small number of objects (e.g., less than 10). Indexing, might result in poor performance especially in cases of images with many objects (e.g., more than 10 or 15) which are very similar to each other.

Input: Query Q, Distances D, D_f, Number k of nearest neighbors (NNs).

Output: The k NNs of Q with regard to D.

1. Decompose Q into $Q_1, Q_2, \ldots Q_n$ subqueries.
2. For each Q_i apply a k-NN query on the SAM with respect to D_f. Each Q_i retrieves *exactly* k images; Compute t_i, their maximum distance D from Q.
3. Compute $t = \min_{i=1}^{n}\{t_i\}$.
4. Apply a range query $D(Q, I) \leq t$ using the algorithm of Figure 8.
5. Output the k images closest to Q.

Figure 9: *Algorithm to retrieve the k NNs of a query with regard to distance D, using a SAM.*

4.3 Nearest Neighbor (NN) Queries

Algorithms for NN searching on SAMs like e.g., [34, 43] return the k most similar images of a Q_i with regard to D_f. The objective is to find the k most similar images of a query Q with regard to $D(Q, I)$. Figure 9 illustrates an algorithm to retrieve the k most similar images of Q using a SAM.

Lemma 3 *The algorithm of Figure 9 retrieves exactly k images.*

Proof 3 *The range query at step 4 retrieves at least the k NNs corresponding to the Q_i with the minimum t_i. Therefore, step 5 accepts at least k images and outputs the best k of them.* □

Lemma 4 *The algorithm of Figure 9 guarantees no false dismissals.*

Proof 4 *Let I be the k-th (last) NN of Q with regard to D. Step 4 retrieved I, that is $D(Q, I) \leq t$ (assumption). Let X be the i-th NN ($i < k$) of Q with regard to D, that is, $D(Q, X) \leq D(Q, I)$. Suppose that the algorithm failed to retrieve X. This is possible only if X did not satisfy the range query at step 4, that is $D(Q, X) > t$. By combining the last two inequalities $D(Q, I) > t$ which contradicts the assumption.*□

The algorithm requires that step 2 retrieves exactly k images, that is exactly k distinct image identifiers I. A conventional NN algorithm (e.g., [34]) would return the k NNs (subimages) I_j of each Q_i which may correspond to less than k distinct image identifiers I (if two or more I_js are derived from the same I). One solution to this problem would be to reinvoke the NN algorithm for more neighbors until k distinct image identifiers are obtained for each Q_i. This approach involves many redundant computations. An alternative would be to apply an incremental k-NN algorithm [43] so that the extra NNs are obtained without having to recompute the first k neighbors. In this work, such an incremental algorithm is not available. The results reported in Section 5 are obtained using the NN algorithm of the DR-tree (a version of the R-tree developed at the Univ. of Maryland) which is a variant of the algorithm by Rousopoulos and Vincent [34]. The experiments demonstrate that even this non-incremental algorithm retrieves the actual k-NNs with almost 100% accuracy.

An alternative to this approach would be to ask a sequence of range queries with successively increasing range. Depending on how successful such a guess can be, this approach will be faster than the above proposed algorithm.

4.3.1 Discussion

The algorithm works even for $n = m = 1$, that is for images and queries with exactly one object. In that respect, the proposed approach can be viewed as a generalization of the NN algorithm of [27] which works only for images with one object.

The NN algorithm calls for the range search algorithm of Figure 8 at step 4. Therefore, NN searching is expected to be slower than range searching, at least for queries retrieving the same number of answers. As noticed in Section 4.2.1, the range search algorithm need not necessarily apply all its n intermediate queries (the algorithm might be faster if less than n intermediate queries are applied). If an optimal value of intermediate subqueries for the range search algorithm has been specified, the same value can also be used by the NN algorithm at step 4 (i.e., this value is 2 in this work).

Response times depend highly on t, the value of the distance tolerance for the range queries at step 4. This value depends on the distance from Q of all the NNs retrieved at step 2. For large values of t, these range queries may become very slow. Both, D and D_f must take values within the same range (e.g., [0,2] in this work). Otherwise, if D takes higher values, the range query

at step 4 might retrieve the whole database. This might happen, for example, if D is computed according to Equation 2. Equation 3 provides the only definition of D that guarantees values of the distance within $[0, 2]$ and can always be used effectively with the proposed algorithm for NN queries.

Notice also that, not all NN queries need to be applied at step 1. The smaller the value of the distance tolerance t at step 3 the faster the algorithm is. The purpose of Step 3 is to compute the smallest possible value of t. The possibility of computing the smallest possible value of t increases as more NN subqueries are applied at step 1. In this work, all the n subqueries are applied. The experiments indicated that even this exhaustive specification of t is very relaxed: Even a small fraction (e.g., 20%) of the actual t computed at step 3 might also be used. This approach trades off a tiny loss in accuracy (e.g., misses less than 10% of the actual NNs) for many times (e.g., 5) faster retrievals.

4.4 Comparison with Previous Work on Image Indexing

Existing work on image indexing by content is mostly influenced from the database paradigm. The main idea is always the same: Extract a fixed number of features from each image (or query) and map these features to points in a multidimensional space. A SAM can then be used to process range and NN queries. However, SAMs (e.g., [10, 34, 18]) do not treat image content directly (e.g., objects are approximated by their minimum bounding rectangles) nor can they treat multiple objects and their interrelationships. The method by Korn et.al. [27] handles both range and NN queries in conjunction with indexing using R-trees, but cannot treat multiple objects and relationships between objects. The method by Petrakis and Faloutsos [1] works for multiple objects and relationships but only for range queries; it achieves fast retrievals by content by exploiting the assumption that each image or query always has a fixed number of "expected" objects (plus one or two unknown objects) and by indexing ARGs with R-trees. The proposed method relaxes these restrictions and treats images and queries with any number of objects. Other methods focusing on spatial relationships (e.g., [15, 33]) assume sequential search of the entire IDB. The method by El-Kwae and Kabuka [44] is a multilevel indexing technique based on bit signatures. The index is compact and fast and supports exact-match queries by image content; range and NN queries are not supported.

A recent contribution by Papadias, Mamoulis and Delis [45] shares some commonalities with the proposed approach in that it is motivated by an effort to avoid the high computational complexity of sequential search for structural queries in image databases. 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 translations. This approach seems to be targeted mainly to geographic information databases where all these requirements are usually met. Its extension to treat images of arbitrary content complexity is non-trivial. In addition, the method does not focus on range or NN queries (i.e., given a query, it is mapped into regions in a multidimensional space and all regions that intersect the query regions are retrieved).

Methods from the Computer Vision research are based on the idea of indexing ARGs using precomputed information. A recent contribution by Segupta and Boyer [46] proposes 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). Similarly, the method by Messmer and Bunke [47] 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 are not designed for answering range or nearest-neighbor queries in image databases, they have not been tested on large datasets storing thousands of graphs and, finally, they are static (i.e., they cannot handle insertions

or deletions). Their emphasis is on the design of efficient algorithms for computing the error correcting isomorphism between a query and set of stored ARGs.

The proposed indexing method is mainly inspired from the database paradigm in that it is intended to provide efficient access to range and NN queries on databases of random ARGs stored in the main memory or on the disk (an issue that is not considered by the above Computer Vision methods). No assumption is made about the content of the stored images or of the queries. The proposed method utilizes existing image matching algorithms (it doesn't propose a new one). In fact, the faster the algorithm which is used for image matching the faster the search of the IDB.

It is an interesting future project to have a thorough comparative experimental evaluation with all the above methods. However, the methods are very different from each other and their implementation is nontrivial.

5 Experiments

The R-tree and the search algorithms are implemented in C while, the image algorithms are implemented in $C++$. The experiments were run on a dedicated SUN Ultra 1 (167MHz) with 128Mbytes main memory running Solaris^{TM}. The dataset of $13,500$ images of [1] was used as a testbed[1]. All images contain between 4 and 8 objects. In all the experiments $f = 5$. The R-tree has page size is 1 Kbyte, it contains 90,000 entries (total number of objects in all the 13,500 images) and required 7.14 Mbytes for storage. The size of the ARG file is 22.8 Mbytes. Both, the ARG file and the R-tree are loaded eventually into the main memory (due to I/O buffering). For the evaluations 20 characteristic queries were created. All results are averages over 20 queries. The queries contain between 4 and 6 objects.

The experiments are designed to demonstrate the *superiority* of the proposed methods for range and NN search in IDBs over sequential scan searching. The times reported correspond to elapsed (wall-clock) times computed using the `time` system call of UNIX. Both elapsed (total) and `system` time are reported.

There are 6 variants of the ARG editing distance corresponding to all possible combinations of Φ and ϕ (see also Section 3.2). In all the experiments below Φ_{max} and ϕ_1 are used. Notice that $\phi_1 \leq \phi_\infty$: Range queries with ϕ_1 specify a smaller distance tolerance for NN search and are faster

[1]The images and the queries are available from `http://www.ced.tuc.gr/~petrakis`

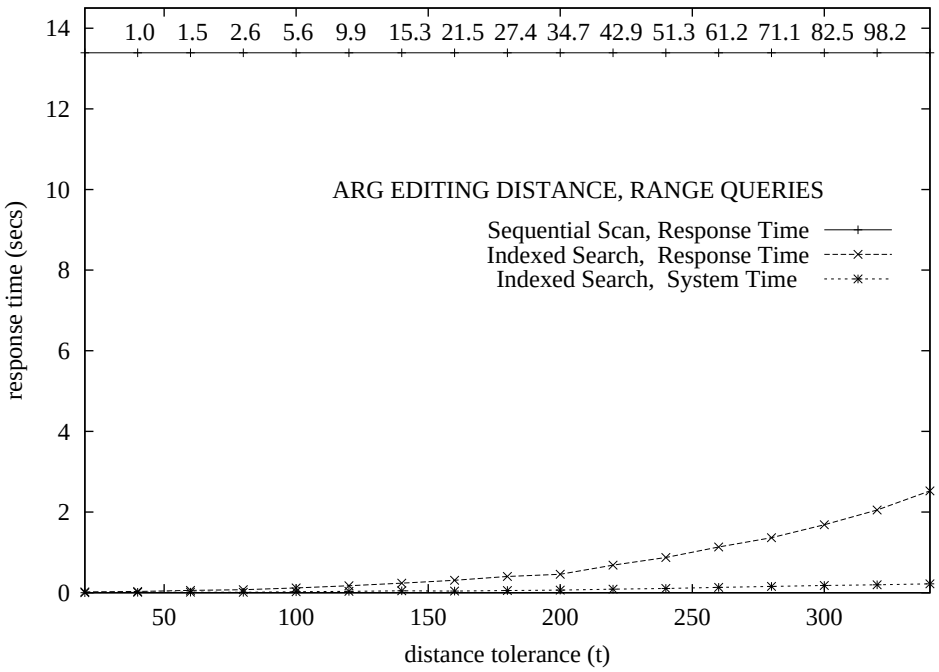

Figure 10: *Average retrieval response time as a function of the distance tolerance t corresponding to (a) sequential search of the database and (b) indexed search on a 5-dimensional R-tree, using the ARG editing distance. The labels denote average numbers of retrieved images.*

than queries with ϕ_∞. The same hold for ϕ_2 and ϕ_∞.

5.1 Range Queries

Figure 10 plots the response time for range queries as a function of the distance tolerance t for both indexed and sequential search using the ARG editing distance. Figure 11 illustrates the same information but using the Hungarian distance. The labels above the sequential method denote average numbers of retrieved images. The proposed method achieves large savings for both definitions of the distance, even for high values of the distance tolerance (e.g., for $t = 200$ the retrieved set is almost 35 images). Even for such large queries the response times is below 3 seconds. Sequential scanning is always above 12 seconds for both methods.

Figure 10 and Figure 11 also illustrate system time. Notice that, most of the response time is cpu (user) time devoted to ARG distance calculations (clean-up). Only a small percentage is system time spent for R-tree and ARG file search (system time).

The shape of the curves is justified as follows: For sequential scanning, the search time is

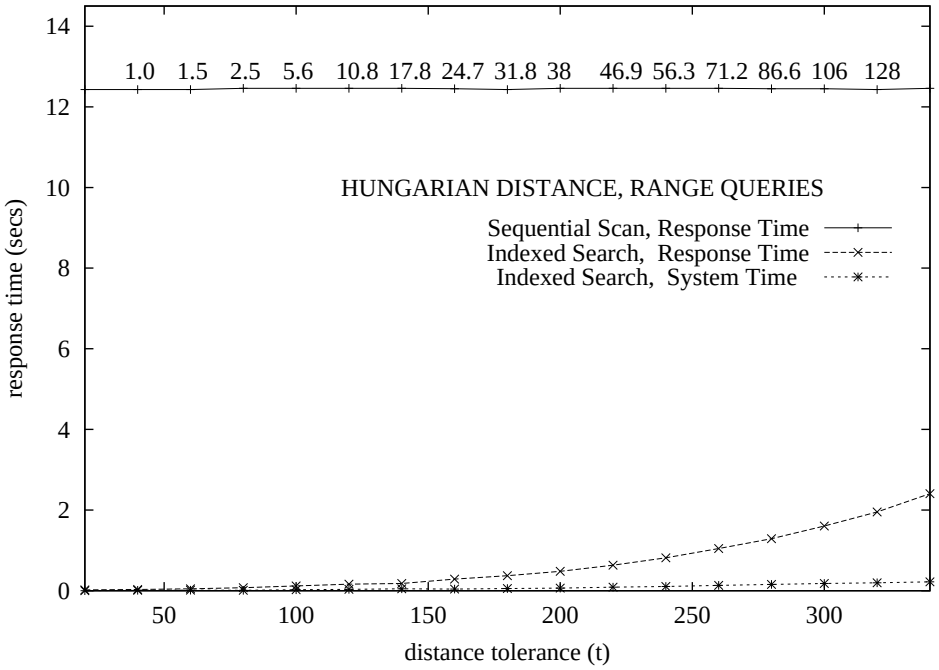

Figure 11: *Average retrieval response time as a function of the distance tolerance t corresponding to (a) sequential search of the database and (b) indexed search on a 5-dimensional R-tree, using the Hungarian distance. The labels denote average numbers of retrieved images.*

always the same regardless of distance tolerance. For the proposed method, the shape of the curve resembles an exponential curve. This is expected because the number of candidate images which are retrieved by D_f increases exponentially with the distance tolerance (i.e., the number of points around the query increases with t^f). The more the retrieved images the slower the retrieval response times are (all these images account for ARG distance calculations in clean-up).

5.2 Nearest Neighbor (NN) Queries

Figure 12 plots the response time as a function of the k, the number of retrieved NNs using the ARG editing distance. Similarly, Figure 13 illustrates the same information but using the Hungarian distance. Again, the plot on the left corresponds to the retrievals with the ARG editing distance and the plot on the right corresponds to retrievals with the Hungarian distance. The time for sequential scanning and system time are also shown. Notice that, response time increases only for $k < 10$ and remains practically constant for greater values of k. Retrieval response time depends on the distance tolerance t which is computed at steps 2-3 of the algorithm of Figure 9. The value

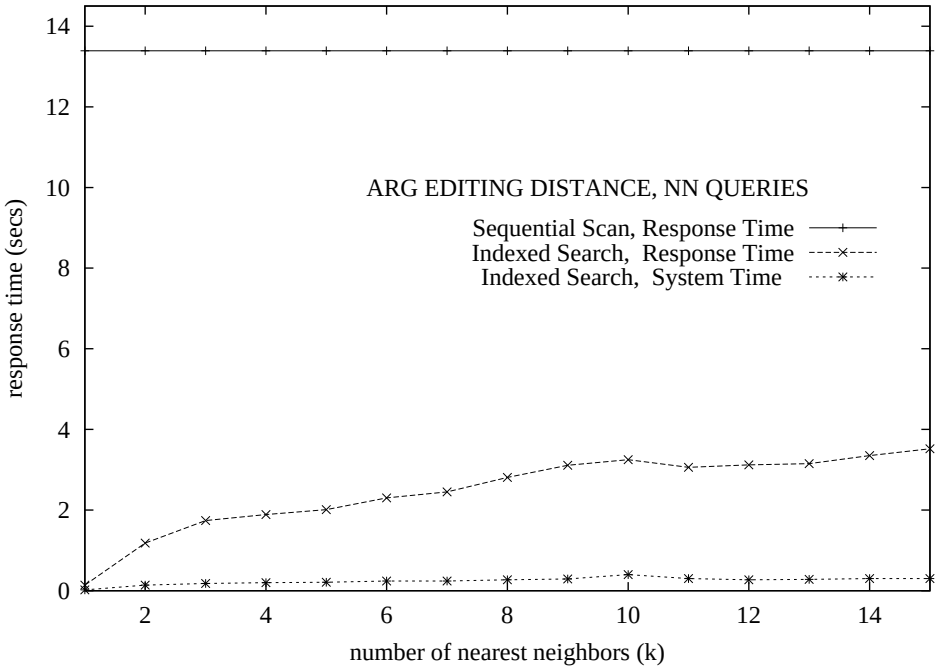

Figure 12: *Average retrieval response time as a function of the number of Nearest Neighbors (NNs) retrieved corresponding to (a) sequential search of the database and (b) indexed search on a 5-dimensional R-tree, using the ARG editing distance.*

computed for small k seems to be also adequate for retrieving even more NNs. This also means that the specification of t by the algorithm is too relaxed and that a much smaller value of t might also be used. The experiments indicated that the method retrieves 90% of the same NNs with the 30% of the value t specified by the algorithm. Retrieval response times for such values of t became more than 5 times faster. Again, system time is only as small percentage of the total response time.

5.3 Discussion

The proposed approach relies on range and NN searching on a SAM and it is particularly well suited for low dimensionality feature vectors (i.e., most SAMs reduce to sequential scanning for more than 10 or 15 dimensions). For high dimensional SAMs it would be beneficial to keep only "useful" attributes (e.g., the most discriminatory attributes) for indexing. An in-depth analysis of such feature reduction methods is outside the scope of this paper. The following outlines two possible solutions: (a) A domain expert may be used to pinpoint the useful attributes, (b) In case of a static (or slowly changing) database, the Karhunen-Loeve (K-L) transform can be applied,

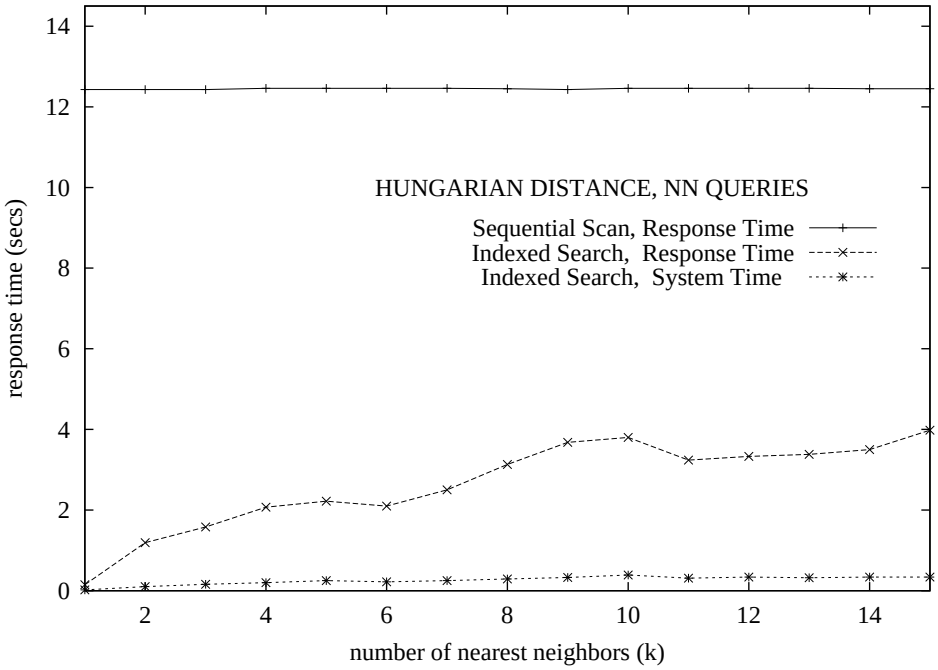

Figure 13: *Average retrieval response time as a function of the number of Nearest Neighbors (NNs) retrieved corresponding to (a) Sequential search of the database and (b) Indexed search on a 5-dimensional R-tree, using the Hungarian distance.*

which, given f attributes, creates f new attributes sorted in "usefulness" order. The first few attributes retain most of the information needed.

An interesting observation is that the same SAM is used to answer queries with both, the ARG editing and the Hungarian distance (the ARG editing distance takes the relationships between regions into account while, the Hungarian distance ignores these relationships).

5.4 Examples of Retrievals

Queries with multiple regions are typically specified by example image (i.e., by drawing a sketch on the screen). This permits even complicated queries with many regions and complex spatial relationships. Figure 14 illustrates a typical query (top left image) specifying 4 regions and 3 qualifying images retrieved by the ARG editing distance method. Regions 1, 2, 3 and 4 in the query image are matched with objects with the same indices in the retrieved images. In this example, for simplicity, all associated regions appear with the same indices. Notice that, a retrieved image (but not the query) may contain extra regions.

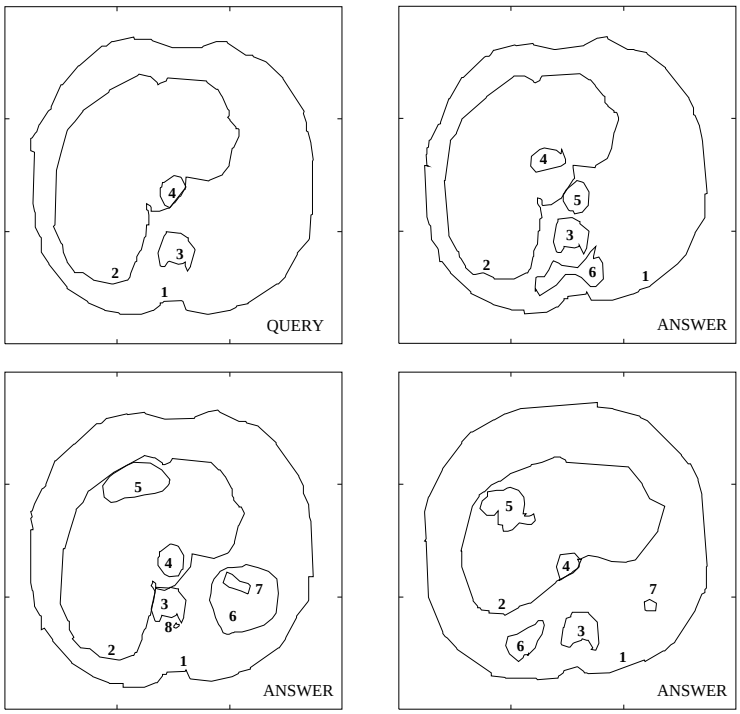

Figure 14: *Example of a query image (top left) and three qualifying images.*

6 Conclusions

The emphasis of this work is on showing how a spatial access method (e.g., an R-tree) can be used to answer range and nearest neighbor queries on data entities which are represented by multiple multidimensional points (i.e., images with multiple regions in this work). Two algorithms are proposed, one for each type of query. Both these algorithms guarantee no false dismissals and no false drops, work with any spatial access method and with any image distance function provided that it satisfies the so called Lower Bounding Principle.

A critical analysis of the performance of range and NN queries is presented and discussed. The results demonstrate a very significant improvement in the retrieval performance of range queries. For NN queries the improvement is not dramatic but can be improved further by introducing certain heuristics in the search process. The proposed method is best suited for "small" images with less than 10 or 15 objects.

Future work includes experimentation with more data types and the investigation for more efficient search algorithms. A very promising direction is the study of data mining algorithms on

the point-transformed images to detect regularities and patterns in geographic and demographic databases.

Acknowledgements

I am grateful to Prof. Dimitris Papadias of the Dept. of Comp. Science at the Univ. of Hong Kong for his insightful comments and to Prof. Christos Faloutsos of the Dept. of Comp. Science at the Carnegie Mellon University for providing me the codes of the DR-tree. I am also grateful to Costas Georgiadis who implemented the image matching algorithms.

References

[1] E. G.M. Petrakis and C. Faloutsos. Similarity Searching in Medical Image Databases. *IEEE Trans. on Knowl. and Data Eng.*, 9(3):435–447, May/June 1997. (http://www.ced.tuc.gr/~petrakis).

[2] N. K. Ratha, K. Karu, S. Chen, and A. K. Jain. A Real Time Matching System for Large Fingerprint Databases. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 18(8):799–813, Aug. 1996.

[3] A. K. Jain and A. Vailaya. Shape-Based Retrieval: A Case Study With Trademark Image Databases. *Pattern Recogn.*, 31(9):1369–13990, 1998.

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

[5] 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 Patt. Anal. and Mach. Intell.*, 22(11):1349–1380, Dec. 2000.

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

[7] 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).

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

[9] 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).

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

[11] E. G.M. Petrakis. Design and Evaluation of Spatial Similarity Approaches for Image Retrieval. *Image and Vision Comp.*, 20(1):59–76, 2002.

[12] S.-K. Chang. *Principles of Pictorial Information Systems Design*. Prentice Hall Intern. Editions, 1989.

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

[14] E. G.M. Petrakis and S. C. Orphanoudakis. Methodology for the Representation, Indexing and Retrieval of Images by Content. *Image and Vision Comp.*, 11(8):504–521, Oct. 1993.

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

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

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

[18] 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.

[19] W. W. Chu, C.-C. Hsu, A. Cardenas, and R. K. Taira. Knowledge-Based Image Retrieval with Spatial and Temporal Constructs. *IEEE Trans. on Knowl. and Data Eng.*, 10(6):872–888, Nov./Dec. 1998.

[20] A. Del Bimbo, M. De Marsico, S. Levialdi, and G. Peritore. Query by Dialog: An Interactive Approach to Pictorial Querying. *Image and Vision Comp.*, 16(8):557–569, June 1998.

[21] M. Das, E. M. Riseman, and B. A. Draper. FOCUS: Searching for Multi-colored Objects in a Diverse Image Database. In *Proc. of IEEE Comp. Soc. Conf. on Comp. Vision*, pages 756–761, 1997.

[22] J. Huang, S. R. Kumar, M.Mitra, W.Zhu, and R. Zabih. Image Indexing Using Color Correlograms. In *Proc. of IEEE Comp. Soc. Conf. on Comp. Vision*, pages 762–768, 1997.

[23] A. Mojsilovic, J. Kovacevic, J. Hu, R. J. Safranek, and S. K. Ganapathy. Matching and Retrieval Based on the Vocabulary and Grammar of Color Patterns. *IEEE Trans. on Image Proc.*, 9(1):38–54, Jan. 2000.

[24] B. M. Mehtre, M. S. Kankanhalli, and W.-F. Lee. Shape Measures for Content Based Image Retrieval: A Comparison. *Info. Proc. and Manag.*, 33(3):319–337, 1997.

[25] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and Robust Retrieval by Shape Content through Curvature Scale Space. In *Proc. of Intern. Workshop on Image DataBases and MultiMedia Search*, pages 35–42, Amsterdam, The Netherlands, 1996. (http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html).

[26] E. Milios and E. G.M. Petrakis. Shape Retrieval Based on Dynamic Programming. *IEEE Trans. on Image Proc.*, 9(1):141–147, Jan. 2000.

[27] P. Korn, N. Sidiropulos, C. Faloutsos, E. Siegel, and Z. Protopapas. Fast and Effective Retrieval of Medical Tumor Shapes. *IEEE Trans. on Knowl. and Data Eng.*, 10(6):889–904, Nov./Dec. 1998.

[28] T. Gevers and A. W. M. Smeulders. PicToSeek: Combining Color and Shape Invariant Features for Image Retrieval. *IEEE Trans. on Image Proc.*, 9(1):102–119, Jan. 2000.

[29] S. Sclaroff. Deformable Prototypes for Encoding Shape Categories in Image Databases. *Pattern Recogn.*, 30(4):627–641, 1997.

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

[31] S.-Y. Lee and F.-J. Hsu. 2D C-String: A New Spatial Knowledge Representation for Image Database Systems. *Pattern Recogn.*, 23(10):1077–1087, 1990.

[32] J. R. Smith and S.-F. Chang. Integrated Spatial and Feature Image Query. *Multimedia Systems*, 7:129–140, 1999. (http://disney.ctr.columbia.edu/safe).

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

[34] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest Neighbor Queries. In *Proc. of ACM-SIGMOD*, pages 71–79, San Jose, CA, March 1995.

[35] T Brinkhoff, H.-P. Kriegel, R. Schneider, and B. Seeger. Multi-Step Processing of Spatial Joins. In *Proc. ACM SIGMOD*, pages 197–208, Minneapolis, MN, May 1994.

[36] V. Gaede and O. Gunther. Multidimensional Access Methods. *ACM Comp. Surveys*, 30(2):170–231, June 1998.

[37] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proc. of ACM SIGMOD*, pages 322–331, Atlantic City, NJ, May 1990.

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

[39] E. G.M. Petrakis, C. Faloutsos, and K.-Ip Lin. ImageMap: An Image Indexing Method Based on Spatial Similarity. *IEEE Trans. on Knowl. and Data Eng.*, 1999. (to appear, http://www.ced.tuc.gr/~petrakis).

[40] W. J. Christmas, J. Kittler, and M. Petrou. Structural Matching in Computer Vision Using Probabilistic Relaxation. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 17(8):749–764, Aug. 1995.

[41] B. T. Messmer and H. Bunke. A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 20(5):493–504, May 1998.

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

[43] G. R. Hjaltason and H. Samet. Distance Browsing in Spatial Databases. *ACM Trans. on Inf. Syst.*, 24(2):265–318, 1999.

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

[45] 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.

[46] K. Segupta and K. L. Boyer. Organizing Large Structural Modelbases. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 17(4):321–332, April 1995.

[47] B. T. Messmer and H. Bunke. Fast Error-Correcting Graph Isomorphism Based on Model Precompilation. Tech. Report IAM-96-012, Univ. of Bern, Switzerland, Sept. 1996. (http://www.iam.unibe.ch/~fkiwww/publications/index.html#technicalreports).

Contents