Design and Evaluation of Spatial Similarity Approaches for Image Retrieval

Euripides G.M. Petrakis
Department of Electronic and Computer Engineering
Technical University of Crete
Chania, Crete, Greece
petrakis@ced.tuc.gr

September 7, 2001

Abstract

Similarity retrieval by spatial image content (i.e., using multiple objects and their relationships in space) is an open problem which has received considerable attention in the literature. The most powerful approaches of spatial image content representation and retrieval are *"Attributed Relational Graphs"* (ARGs) and *"Symbolic Projections"* (e.g., 2D Strings). In this work, a framework is proposed for studying the performance of such spatial similarity approaches in Image DataBases (IDBs). The classical ARG and 2D string matching methods are evaluated. Several variants of ARG and 2D string methods for improving their accuracy and speeding-up their time responses are also proposed and tested. A critical analysis of the performance of all these methods is presented. The analysis indicates that, in retrieving images by spatial content, retrieval response time and accuracy are traded-off.

Key-Words: image database, image retrieval, spatial relationships, attributed relational graphs, 2D strings.

1 Introduction

Several approaches to the problem of content-based image management have been proposed and some have been implemented on research prototypes and commercial systems [1, 2, 3, 4, 5]. Focusing mainly on color, texture and shape, the work referred to above does not show how to handle multiple objects or regions in spatial queries, nor their inter-relationships. Typically, spatial queries are specified by example image. Queries by example permits even complicated queries: The user may specify several objects with (possibly) complex shapes and complex relationships. 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 (i.e., spatial relationships) in the two images.

There are two general goals common to all Image DataBase (IDB) systems:

Effectiveness: An IDB system must treat images and queries of arbitrary content complexity 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: It must be *fast* to meet the speed requirements many IDB applications.

This work emphasizes on spatial image content and treats both these issues. In particular, this work

- Presents a critical analysis of the performance of the most powerful methods of spatial image content representation and matching, such as the well established "editing distance" on ARGs, the so called "Hungarian Method" for graph matching and several 2D string matching methods.
- Proposes various extensions to the classical ARG and 2D string methods which demonstrate better performance than their original counterparts.
- Demonstrates that in retrieving images by spatial content, retrieval response times and accuracy are traded-off.

The performance of several ARG and 2D string matching methods is evaluated using the dataset of 13,500 synthetic images of [6]. The accuracy of all candidate methods is evaluated based on human relevance judgments following a well established methodology from the information retrieval field. This set of experiments required over than 20,000 relevance judgments by an independent referee. The focus of this work in not on choosing a good set of features for a particular type of images (medical images in this work) or representation but, on choosing the more effective (i.e., accurate) image content representation and retrieval method for a given set of features (the features are common to all methods under consideration). Average measurements of the retrieval response times of all methods are also taken and used to demonstrate the trade-off between accuracy and speed.

It is assumed that each image has been segmented to more than one objects or regions. Image segmentation is a difficult problem which is outside the scope of this paper. For the purposes of this work, the images are given in the desired segmented form. Figure 1 shows an example of an original grey-level image (left) and its corresponding segmented form (right). In this example, each object or region is represented by its surrounding contour. These segmented forms are stored in the IDB together with the original images. They can be used for browsing the contents of the IDB and for computing a variety of image features specific to a particular image representation. These representations are then used to search the IDB and to determine which images satisfy the query selection criteria.

The rest of this paper is organized as follows: A review of the related work is presented in Section 2. Attributed Relational Graphs and 2D strings are presented in Section 3 and Section 4 respectively. The evaluation method is discussed in Section 5. Experimental results are presented and discussed in Section 6 followed by conclusions in Section 7.

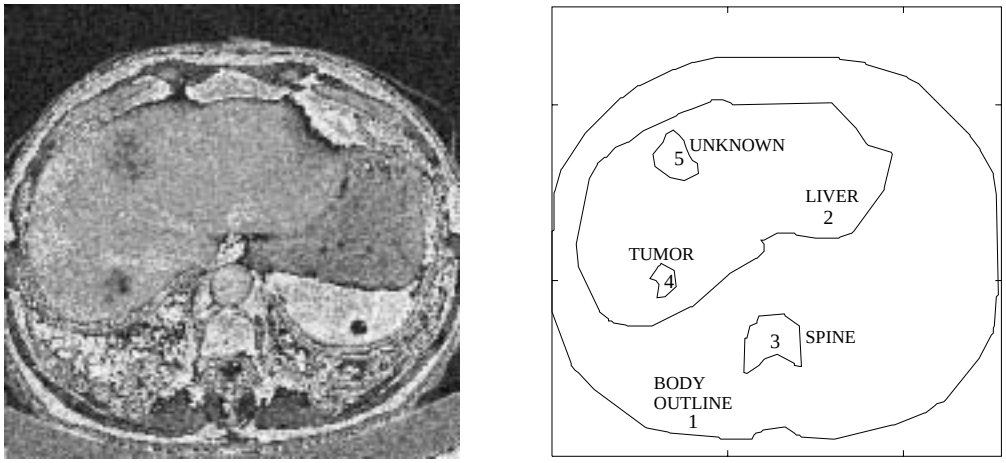


Figure 1: *Example of an original grey-level image (left) and its segmented form (right).*

2 Related Work

Advances mainly in the areas of Databases and Computer Vision research resulted in methods which can be used for image archiving, retrieval and IDB design work. Recent surveys have been published in [7, 8].

2.1 IDB Methods and Systems:

In the Virage system [4], image content is given primarily in terms of properties of color and texture. The QBIC system of IBM [1] utilizes a retrieval method for queries on color, texture and shape. The Photobook [2], the system developed at the MIT Media Lab, supports queries by image content in conjunction with text queries. VisualSEEk [3] 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 the WWW. Chu et.al. [9] 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., [10, 11]). Additional work on retrieval by color and texture, includes (among others) the work by Huang et.al. [12], who suggests using the color image attributes as measurements of image content in image databases. Mojsilovic et.al. [13] 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.

A separate category of methods emphasizes on retrieval by shape content. The effectiveness of shape methods in conjunction with color features is investigated by Jain and Vailaya [14] using a dataset of trademark images. SQUID [15] supports retrievals on a dataset of marine life species. Milios and Petrakis [16, 17] demonstrate the superiority of their multiscale dynamic programming matching methods over Fourier and moment-based methods using static hand gesture shapes and the

dataset of SQUID. Korn et.al. [18] 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. Del Bimbo et.al. [11] 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. PicToSeek [5] combines shape and color features for shape retrieval.

2.2 Retrieval by Spatial Similarity

In retrieval by spatial image content, not only the shape, color and texture properties of individual image regions must be similar, but they must have the same arrangement (spatial relationships). The textbook approaches to treat this information in IDB systems are 2D strings and Attributed Relational Graphs (ARGs).

2D strings [19] and their variants [20] are examples of work focusing mainly on spatial relationships. 2D strings constitute an efficient representation of certain types of spatial relationships (i.e., "left/right" and "below/above") and object properties and provide low complexity (i.e., polynomial) matching in image databases. However, 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) [21]. 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 [22] handle overlapping objects with complex shapes but they are not as simple and compact as the original 2D strings nor can they be produced very efficiently. Besides, the matching of 2D C strings has exponential time complexity. Finally, the so called "expanded 2D strings" [23] treat more object properties and relationships while, 2D string matching remains polynomial in time. The problem of indexing 2D strings in IDBs is also addressed in [21, 23].

ARGs are the most general image content representation method but, they have exponential time complexity for matching. 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. Image matching is treated as an error-correcting ARG matching problem [24]. Traditionally, this can be formulated as a tree search problem which can be solved by an $A^{\star}$ algorithm [25]. This is the representative of a large class of ARG matching methods relying on tree searching [26, 27, 28, 29]. All these methods have exponential time complexity.

Approximate ARG matching methods with lower time complexity do exist but they are not guaranteed to find the optimal solution. Almohamad and Duffuaa [30] propose a method based on linear programming [30]. Christmas, Kittler and Petrou propose a method based on probabilistic relaxation [31]. Probabilistic relaxation methods (see also [32, 33]) are very popular mainly because of their low polynomial time complexity of matching. Their disadvantage is that they may get trapped in local minima and miss the optimal solution. The graduated assignment method by Gold and Rangarajan [34] is a substantial improvement of the previous type of methods that promises more accurate results.

A recent contribution by Messmer and Bunke [35] proposes a polynomial time algorithm for error-correcting matching of labeled ARGs. The main idea behind their approach is to decompose all ARGs in a database into subgraphs and store these subgraphs in a tree-like structure where the common subgraphs of all ARGs are represented only once. These subgraphs are then matched only once with the query graph, thus saving time when the ARG database is searched. Similarly, Segupta and Boyer [36] propose a hierarchical organization of graphs, where the root graph consists of different distinct subgraphs derived from all stored graphs. However, the matching of a query with the root graph may become very time consuming. In addition, their method is approximate (i.e., may miss an actual match). Shapiro and Haralick [37] propose a hierarchical clustering of ARGs into classes of similar ARGs. The main advantage of all these methods is that they are particularly well suited for searching in image databases since they avoid the matching of the query with every stored ARG. Their main disadvantage is that they require an expensive preprocessing step for building an ARG index structure. In addition, not all of them are optimal (like e.g., [36]), others are restricted to special kind of ARGs (e.g., labeled ARGs [35]), they may require exponential space (e.g., [35]) or can't guarantee low complexity matching (e.g., [36]).

The method by Gudivada and Raghavan [38] emphasizes on special types of image properties (e.g., relative orientation between objects) for the representation of image content (see also Section 3.4). The method by El-Kwae and Kabuka [39] 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. Compared with traditional ARG matching, these two 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 makes no assumption and computes the best association between objects in the two images by taking also their their relationships into account.

To speed-up retrieval response times the stored ARGs can be indexed using Spatial Access Methods (SAMs). Petrakis and Faloutsos [6] transform ARGs to multidimensional points which are indexed by an R-tree. They achieved fast retrievals by spatial image content by exploiting the assumption that each image always has a fixed number of "expected" objects or regions. This retrieval method did not allow for missing or extra regions in images and required that at least some of the regions are labeled and present in every image. These restrictions are relaxed in [40] but, retrieval is approximate. In Papadias, Mamoulis and Delis [41], 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. Additional work on indexing and retrieval based on spatial similarity, includes the method by El-Kwae and Kabuka [42], a multilevel indexing technique based on bit signatures.

3 Attributed Relational Graphs (ARGs)

Figure 2 illustrates the ARG computed to the image of Figure 1. Nodes correspond to regions and arcs correspond to relationships between regions. The arcs are directed from the outer to the contained regions (e.g., from "body outline" to the remaining objects) but their direction also depends on object labels (e.g., from the most common objects like "spine" to the remaining objects).

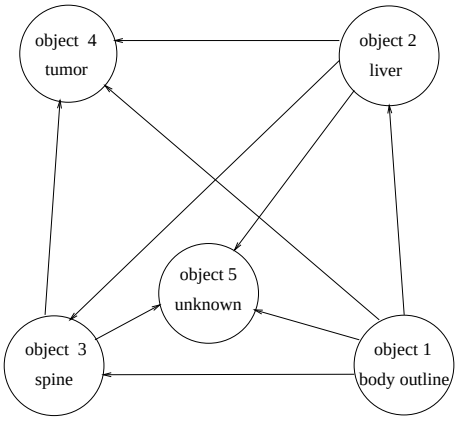


Figure 2: *Example ARG.*

Both nodes and arcs are labeled by attributes corresponding to properties (features) of objects and relationships respectively. A set of features that has been used successfully [21, 6, 40] 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 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 $[0, 2]$. In particular, 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 $[0, 1]$ and is multiplied by 2. Relative distances are divided by the half of the maximum relative distance between any two regions. 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.

3.1 ARG Editing Distance

Matching a query and a stored ARG is treated as an *"error-correcting subgraph isomorphism"* problem [24]: Given their graphs Q (query) and I (model or reference ARG) there is always a sequence of "edit" operations (or corrections) $S(I) = \delta_k(\delta_{l-1}(\ldots\delta_2(\delta_1)\ldots))$ 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 in I (equivalently insertions in 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 with the minimum cost. More formally, the distance $D(Q, I)$ between a query Q and a model graph I is defined as

$$D(Q, I) = \min_{S(I)} \{F(S(I))\} = \min_{S(I)} \{F(f(\delta_k), f(\delta_{l-1}), \ldots f(\delta_1))\}\,, \tag{1}$$

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

$$F_{sum} = F(S(I)) = \sum_{i=1}^{k} f(\delta_i). \tag{2}$$

Alternatively, F can be defined as the maximum of the above f costs:

$$F_{max} = F(S(I)) = \max_{i=1}^{k} \{f(\delta_i)\}\,. \tag{3}$$

If nodes and edges are represented by attribute vectors (as it happens in this work), f 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

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

where p is the order of the metric. For $p = 1$ and $p = 2$ the Manhattan f_1 (city-block) and the Euclidean f_2 distances respectively are obtained. If $p \to \infty$, the Chebeychev distance f_∞ is obtained:

$$f_\infty = f(\delta) = \max_{1\leq i\leq w} \{|z_i - z'_i|\}\,. \tag{5}$$

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 ∞).

3.2 Matching Algorithm

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. Traditionally, this can be formulated as a tree search problem which can be solved by an $A^\star$ algorithm [25]. This is the method used in this work. In the following, the ARG at the left of Figure 3 (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 4. 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 4 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 the nodes currently on the path.

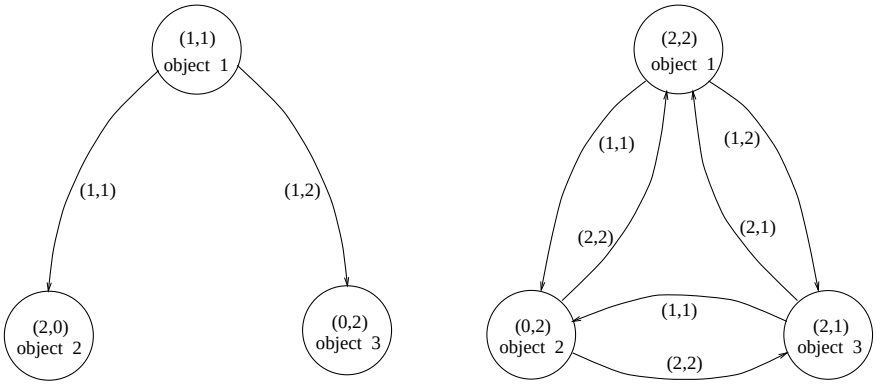


Figure 3: *Example query ARG (left) and reference ARG (right).*

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.

The edit operations for each embedding are recorded and their costs are accumulated in a meaningful way. In Figure 4, node and edge matching costs are computed according 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).

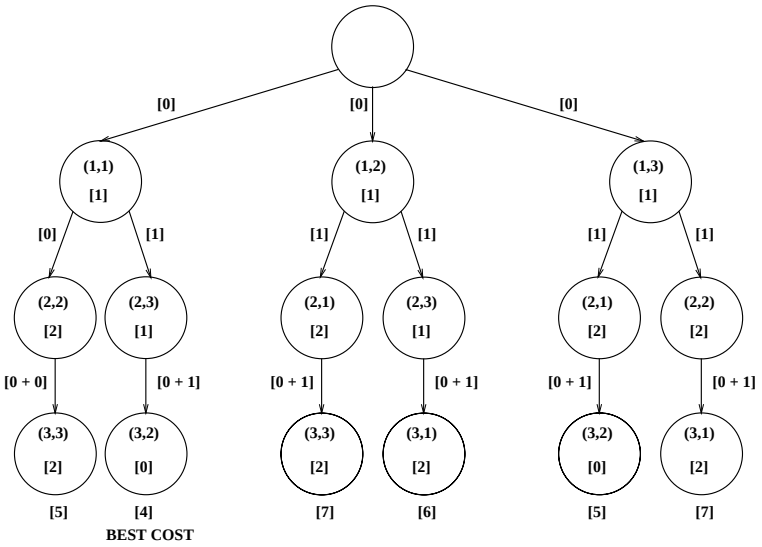


Figure 4: *Matching tree.*

The method for ARG matching referred to above finds the optimal solution but it has exponential time (and space) complexity in the worst case: Each branch of the state-space tree corresponds to a possible mapping of the nodes of Q to the nodes of I. The number of such mappings (equivalently the memory to store the state space tree and the time to search the tree) increases exponentially with the number of nodes in the query and the model ARGs.

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.

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 5 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 [43]. Solid lines in Figure 5 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.

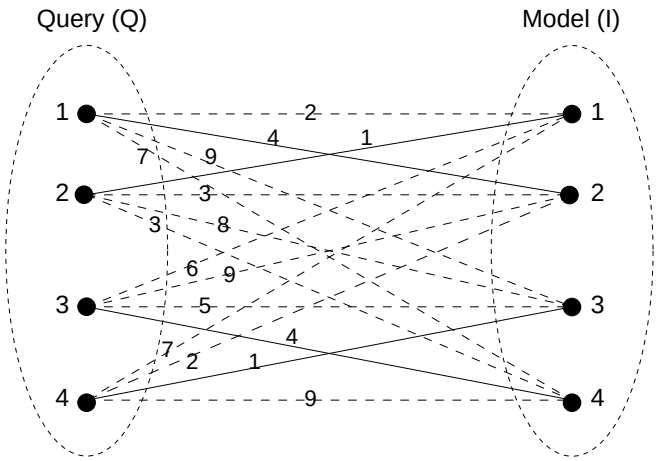


Figure 5: *Image matching as an assignment problem.*

The above formulation assumes that the two ARGs have the same number of nodes. However, if Q contains fewer nodes that I, 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).

3.4 Hungarian Plus

The Hungarian method shows how to find the best association between the objects in a query and a database image but without handling their relationships. Gudivada and Raghavan [38] assume that such an association is given and they compute the cost of this association by taking the differences of the relative orientations between all pairs of matched objects in the two images. The cost $D(Q,I)$ between a query Q and a model image I is computed as

$$D(Q,I) = 1 - \sum_{i,j \leq [1,n]} \frac{1}{n(n-1)/2} \frac{1 + cos(\theta_{i,j} - \theta_{\mathcal{F}(i),\mathcal{F}(j)})}{2}, \tag{8}$$

where $\theta_{i,j}$ is the relative orientation between objects i, j in Q and $\theta_{\mathcal{F}(i),\mathcal{F}(j)}$ is the relative orientation between their associated objects $\mathcal{F}(i)$ and $\mathcal{F}(j)$ in I. This formulae computes the summation of all such differences and then takes their average by normalizing with the number of relationships which is $n(n-1)/2$.

Notice that, [38] doesn't show how to compute a mapping $\mathcal{F}()$. In the present work, the Hungarian method is used to compute this mapping. Then, the cost of matching two images is computed according to Equation 8. This method is called henceforth *"Hungarian Plus"*. A similar approach is discussed in [44].

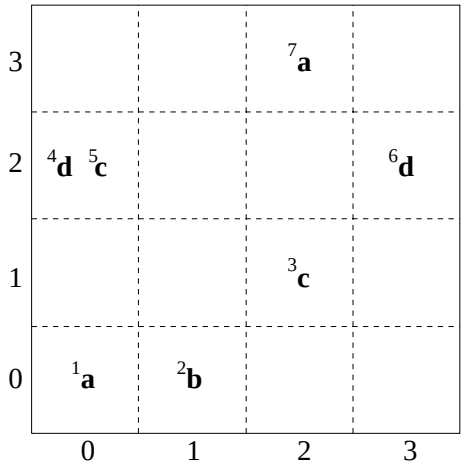
4 Symbolic Projections

2D strings [19] is one of a few methods originally designed for use in an IDB environment. In [45, 23] 2D strings are extended to handle more object properties and relationships. The basic 2D string matching algorithms are also extended to handle the new 2D string representations called "expanded 2D strings".

4.1 2D Strings

To produce the 2D string representation of a given image, the image is segmented and the positions of all objects (i.e. their center of mass) are projected along the x and y directions. By taking the objects from left to right and from below to above, two one-dimensional strings are obtained forming the 2D string representation of the image. The objects themselves are represented by values corresponding to classes or names. For example, the objects contained in the image of Figure 6 can be of one of 4 classes, namely a, b, c and d, and are numbered from 1 to 7. Their indices are shown on the left of their classes.

The symbol "$<$" denotes the "left/right" relationships in string u and the "below/above" relationships in string v. Objects having the same x or y projection can be written in any order. For example, objects 1, 4 and 5 have the same x projection and can be written as acd or cad or dca etc. 2D strings may take various forms namely "reduced", "augmented" etc. To obtain Ohe augmented 2D string, each object in string v is represented by its position in string u. For example, object a is first in u and it is substituted by 1 in v, object b is fourth in string u and it is substituted by 4 in v etc.



Augmented 2D string:

$$(u, v) = (adc < b < ca < d,\ 14 < 5 < 237 < 6)$$

Figure 6: *A symbolic image (left) and its corresponding augmented 2D string (right).*

Notice that, 2D strings are not always tolerant to small variations of object property values (e.g., for objects being close to each other or close to the grid lines) and may become unstable: The 2D string representations can be significantly different even if the images look similar.

4.2 Expanded 2D Strings

A more convenient string encoding of image content is introduced in [23]. It is capable of accommodating in separate one-dimensional strings the ranks (projections) of all objects along the horizontal and vertical directions (strings x and y respectively), the inclusion relationships between all objects (string w) and the size, perimeter, roundness and orientation properties of the objects contained in an image (strings s, p, r and o respectively). Two ordering criteria are also introduced. The first ordering criterion can be used even if the images are scaled with respect to each other while, the second ordering criterion can be used only if the images are at a fixed scale. This ordering information is encoded by string z:

Criterion 1: The objects (in fact their indices) in string z are ordered according to the x coordinates of their centers of gravity. If they have the same x coordinates they are ordered according to their y coordinates. Two or more objects may not be assigned the same position (rank). They are assigned different ranks even if they are very close to each other.

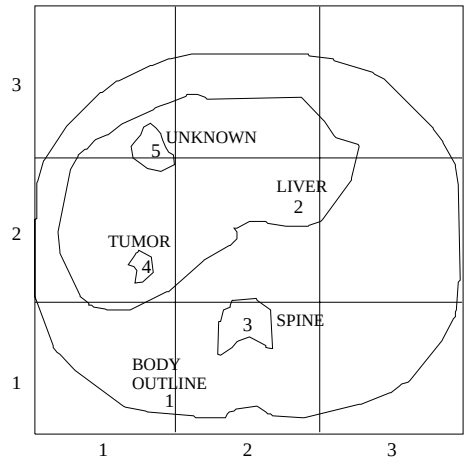
Criterion 2: A criterion of distance is used in assigning ranks to objects, thus the dependence on the scale. Objects are taken on a $M \times N$ rectangular grid (M and N are user defined). They are allowed to share the same position and have the same rank if they are close to each other. Their position is considered to be the grid cell containing their centers of gravity. Objects in string z are ordered based on their horizontal projections (which take values in $[1, M]$). If they have the same horizontal projections, they are ordered according to their vertical projections (which take values in $[1, N]$). This criterion results into more stable representations than criterion 1 (i.e., small variations in object positions are less likely to change their ranks).

The string representation of a given image takes the form (z, x, y, w, s, p, r, o). The exact form of the above representation depends on the order of objects in string z that is, on the ordering criterion applied. Figure 7 shows the expanded 2D strings computed to the image of Figure 1. In each string, the value at position i refers to object $z[i]$. In string w, $w[i]$ denotes the index in z of the object which includes object $z[i]$. For example, $w[1] = 3$ since object $z[1] = 4$ is included by object 2 which is third in z ($z[3] = 2$). In Figure 7, $w[5] = 5$ since object $z[5]$ is not included by any object. All object attributes take values in a range $[1, q]$ where q is user defined ($q = 3$ in this work).

Notice that, not all features may by represented in string form; examples of such features are relative orientation and relative distance. Notice also that, 2D strings can be considered as special cases of expanded 2D strings: They are derived from expanded 2D strings by applying the second ordering criterion, by omitting string w and by keeping only one from the s, p, r and o (usually s).

4.3 2D String Matching

The similarity between two images (e.g., a query and a model image) whose content is represented by two-dimensional strings is treated as a string matching problem. Every object in the query image has to be associated with at least one object (i.e., an object having the same name or class) in a model image and the matched objects in these two images must have exactly the same relationships.



Expanded 2D string	criterion 1	criterion 2
z: order	(4 5 2 3 1)	(4 5 3 2 1)
x: horizontal rank	(1 2 3 4 5)	(1 1 2 2 2)
y: vertical rank	(2 5 4 1 3)	(2 3 1 2 2)
w: relative position	(3 3 5 5 5)	(4 4 5 5 5)
s: area	(1 1 1 1 3)	(1 1 1 1 3)
p: perimeter	(1 1 2 1 3)	(1 1 1 2 3)
r: roundness	(2 2 2 3 3)	(2 2 3 2 3)
o: orientation	(2 3 1 1 1)	(2 3 1 1 1)

Figure 7: *Examples of expanded 2D string representations.*

Three algorithms for 2D string matching and three types of similarity criteria, namely type 0, type 1 and type 2 respectively are defined in [19]. The first algorithm, referred to as $2DmatchA$, is a more general one and may be used with all types of similarity criteria. The other two, referred to as $2Dquery1$ and $2Dquery2$ respectively, have lower time complexity than $2DmatchA$ but, they may be used with only the first two similarity criteria respectively. They produce exactly the same results with $2DmatchA$, only faster.

Type 0 similarity is more general than both type 1 and type 2 similarity (i.e., if two images are type 0 similar, they are also type 1 and type 2 similar). Type 0 similarity differs from type 1 similarity in that it allows for objects in the query image to have the same ranks either on the horizontal or on the vertical axis. Type 2 similarity requires that all query objects exist within the model image and that there are no extra model objects between their associated model objects. The last requirement is relaxed with type 1 and type 0 similarity.

The original 2D string matching algorithms are, in some cases, inexact. The correct versions of these algorithms and their extensions that work with expanded 2D strings can be found in [45].

5 Evaluation Method

Two images are considered similar if they contain similar objects in similar spatial relationships. If the objects in the two images are significantly different, the two images are considered dissimilar even if they have similar relationships. Figure 8 illustrates a typical query (top left image) specifying 4 objects and 3 qualifying images. In this example, for simplicity, objects 1, 2, 3 and 4 in the query image are matched with objects with the same indices in the retrieved images. Notice that, a retrieved image (but not the query) may contain extra objects.

The evaluation is based on human relevance judgments by an independent human referee. For each method, the referee inspected the answers of each query and, for each answer, judged if it is similar to the query or not. This is a highly subjective process. Two or more methods may retrieve

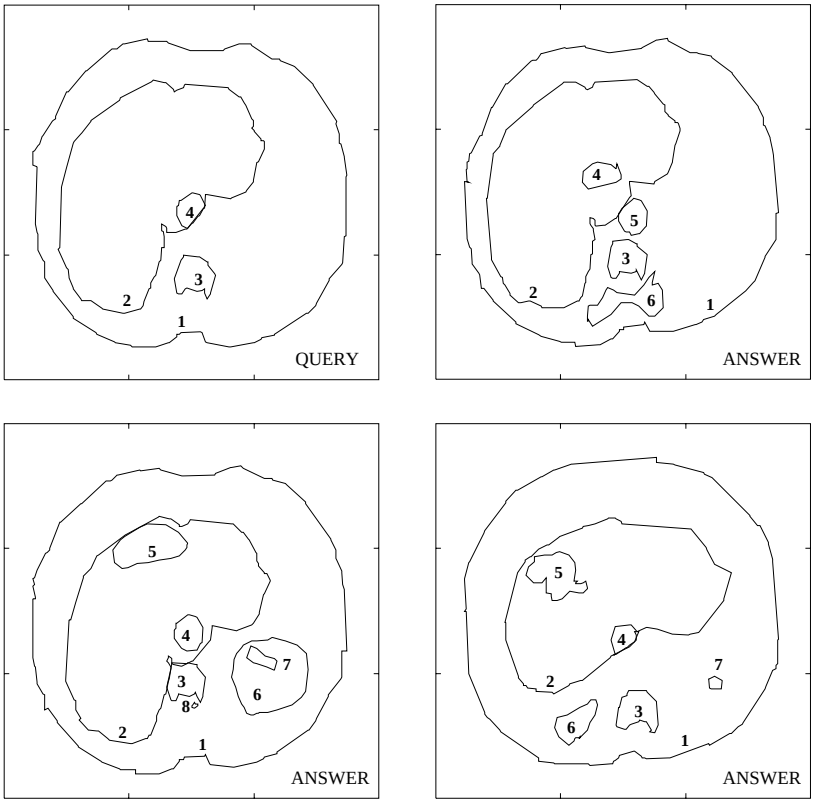


Figure 8: *Example of a query image (top left) and three qualifying images.*

the same answer for the same query, but the same answer may not be recognized as similar when it is retrieved by different methods. To be fair, the evaluations must be consistent. To achieve consistency, a query and a retrieved image are considered as similar if they are taken as similar by at least one method.

To evaluate the effectiveness of each candidate method, the following quantities are computed:

Precision that is, the percentage of qualifying (relevant) images retrieved with respect to the total number of retrieved images.

Recall that is, the percentage of qualifying images retrieved with respect to the total number of images in the database. It is practically impossible to compare every query with each database image. To compute recall, for each query, the answers obtained by all candidate methods are merged and this set is considered to contain the total number of correct answers. This is a valid sampling method, known as *"pooling method"*, which has been used extensively in the text retrieval literature [46]. This method does not allow for absolute judgments such as "method A misses 10% of the total relevant answers in the database". It provides, however, a fair basis for comparisons between methods allowing judgments such as "method A returns 5% fewer correct answers than method B".

Ranking quality R_{norm} which computes the differences between the ranking of the results obtained

by a method and by a human referee [47]. The higher the value of R_{norm} the better the ranking quality of a method, that is, the method retrieves the qualifying (similar) entries before the non-qualifying ones. R_{norm} is computed as follows:

1. The answers of the candidate method are evaluated (i.e., each answer is judged as qualifying or not qualifying).
2. Each answer is assigned a "rank" which equals its order within the answer set (i.e., the first answer has rank 1, the second answer has rank 2 and so on).
3. These answers are taken in pairs such that (a) Only pairs with one qualifying and one non-qualifying answer are taken and (b) In each pair, the qualifying entry is first and the non-qualifying entry is second.
4. The relative ranks of the answers in each pair are examined and R_{norm} is computed as follows:

$$R_{norm} = \begin{cases} \frac{1}{2}(1 + \frac{S^+ - S^-}{S^+_{max}}) & \text{if } S^+_{max} > 0; \\ 1 & \text{otherwise.} \end{cases} \quad (9)$$

   S^+ is the number of correctly ranked pairs (i.e., the qualifying entry has higher rank), S^- is the number of the erroneously ranked pairs (i.e., the method assigned higher rank to the non-qualifying entry) and S^+_{max} is the total number of ranked pairs.

In the following, a *precision-recall* diagram is presented for each experiment. The horizontal axis in such a diagram corresponds to the measured recall while, the vertical axis corresponds to precision. Each method is represented by a curve. Each query retrieves the best 50 answers (best matches) and each point in a curve is the average over 20 queries. Precision and recall values are computed after each answer (from 1 to 50) and therefore, each curve contains exactly 50 points. The top-left point of a precision/recall curve corresponds to the precision/recall values for the best answer or best match (which has rank 1) while, the bottom right point corresponds to the precision/recall values for the entire answer set.

A method is better than another if it achieves better precision and recall. It is possible for two (or more) precision-recall curves to cross-over. This means that one of the methods performs better for small answers (containing less answers than the number of points up to the cross-section) while, the other performs better for larger answer sets. The method achieving higher precision and recall for large answer sets is considered to be the best method (the typical users retrieve more than 10 or 20 images on the average).

Precision and recall are considered to be the standard criteria for evaluating the accuracy of retrieval methods and they have been used extensively within the context of the text retrieval research [46]. Ranking quality is rarely used. Recently it has been adopted for the evaluation of spatial similarity methods [38, 39]. It characterizes the ability of a method to retrieve the correct answers (the answers that the users consider similar to the queries) before the incorrect ones, a desirable feature which, however, cannot show which method is more accurate (i.e., as it is shown in the experiments, the method with the highest ranking quality is not always the most accurate one). In this work, rank-

ing quality is used to distinguish the more effective method among methods with similar precision and recall.

Measurements of the average retrieval response times are also taken. However, the emphasis of this work is not on response times. The time performance of the methods considered in this work have been studied extensively in the literature: [24] reports extensive comparisons of the speed of many ARG matching methods. Additional results with ARG matching and assignment methods can be found in [34, 44, 6] while, the time performance of the 2D strings matching methods considered in this work has been studied in [23]. The results presented in this work confirm (once more) that ARG matching is the slowest method. However, some variants of the ARG matching method have been proven to be fast enough and can be considered as potential alternatives to the less accurate (but faster) polynomial time methods.

6 Experiments

The methods are implemented in C and $C++$ and were run on a dedicated SUN Ultra 1 (167MHz) running Solaris^{TM}. As a testbed, the dataset of $13,500$ synthetic segmented images is used [1] of [6]. For the purposes of the evaluation 20 characteristic queries are created. All images and the queries contain between 4 and 8 objects. Each query retrieves the best 50 answers.

The experiments are designed to demonstrate the:

Efficiency (i.e., speed) of each candidate method. The times reported correspond to elapsed (wall-clock) times computed using the `time` system call of UNIX. These are times for database search; times for image display are not reported.

Effectiveness (i.e., accuracy) of each candidate method. The best method is selected based on measurements of precision, recall and ranking quality.

The competitor methods are:

1. *ARG editing distance:* There exist six variants of the ARG editing distance corresponding to all possible combinations of F and f: f can be defined as a Manhatan, Euclidean or a Chebeychev (maximum) vector distance and F can be defined as a summation or as a maximum operation on the f values.

2. *Hungarian Method:* There are three variants corresponding to the three possible definitions of C (i.e., Manhatan, Euclidean and maximum distance).

3. *Hungarian Plus:* The method of [38] in combination with the Hungarian method. There are three variants of this method corresponding to the three variants of the Hungarian method above.

4. *2D Strings:* This is the original method by Chang et.al. [19] There are three variants of this method corresponding to type 0, type 1 and type 2 matching (similarity) respectively.

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

5. *Expanded 2D Strings:* There are six variants of this method corresponding to type 0, type 1 and type 2 matching in combination with the two ordering criteria.

6.1 Efficiency

The purpose of this set of experiments is to (a) Identify the faster method and (b) Identify the faster variant from each category of methods. Table 1 shows the average retrieval response times obtained by all methods grouped by category. These are times for sequential search of the entire database.

ARG Editing	Hungarian	Hungarian Plus	2D Strings	Expanded 2D Strings
F: Maximum f: Manhatan 16.2 sec	C: Manhatan 13.2 sec	C:Manhatan 16.3 sec	2DmatchA type 0 4.0 sec	criterion 1 1.8 sec
F: Maximum f: Euclidean 16.0 sec	C: Euclidean 13.1 sec	C: Euclidean 16.3 sec	2DmatchA type 1 2.1 sec	criterion 2 1.9 sec
F: Maximum f: Maximum 16.6 sec	C: Maximum 13.5 sec	C: Maximum 16.6 sec	2DmatchA type 2 2.0 sec	
F: Summation f: Manhatan 23.2 sec			2Dquery1 type 1 0.9 sec	
F: Summation f: Euclidean 23.0 sec			2Dquery2 type 2 0.8 sec	
F: Summation f: Maximum 23.6 sec				

Table 1: *Average retrieval response times (in seconds) for sequential search of the entire database.*

2D string methods are obviously the fastest. This is because of their polynomial time complexity of matching. $2Dquery1$ and $2Dquery2$ are faster than $2DmatchA$ because they have even lower time complexity [19]. Notice that, $2DmatchA$ is faster for type 1 and type 2 matching than it is for type 0 matching: The matching algorithm utilizes termination conditions during run-time which are satisfied faster for type 1 and type 2 matching than for type 0 matching.

The version of $2DmatchA$ from [45] capable of handling expanded 2D strings is applied. The expanded 2D strings yielded even faster retrieval response times: 2D string matching is initialized by matching lists (formed by the set of model objects matching each query object without taking their relationships into account). The more object properties used, the shorter the matching lists are and matching is faster. In addition, when more relationships are used, the termination conditions are

satisfied faster.

The ARG editing distance is the slowest method. An interesting observation is that ARG matching with F : maximum, is much faster than ARG matching with F : summation. This is because the former, yields much lower values of the total distance thus resulting in faster pruning of the state-space tree. When the distance takes large values (as it happens with F : summation) the paths in a state-space tree delay to be rejected (the search paths are longer) thus increasing response time. Therefore, the ARG editing distance with F : maximum is more efficient for database search.

Hungarian methods are always faster than ARG matching because of their polynomial time complexity (i.e., ARG matching has, in general, exponential time and space complexity). Hungarian Plus, on the other hand, is slower than the plain Hungarian method because it requires additional processing for the computation of the spatial relationships.

6.2 Effectiveness

The purpose of this set of experiments is to (a) Identify the most accurate variant from each category and (b) Identify the best method from all categories of methods.

ARG Editing Distance: Figure 9 illustrates the precision/recall diagram of the three variants of this family of methods. Matching with F : summation, performs much better than matching with F : maximum, achieving up to 20% better better recall and up to 10% better precision than any other variant. The reason for this behavior is that F : maximum, results in large values of the total distance (compared with the distances between the same query and other images in the database) if the matching of a single node or edge yields a large value. However, this doesn't necessarily mean that the images are dissimilar overall. This is less likely to happen with F : summation: All node and edge costs are accumulated and the matching of single nodes or edges is less likely to result in large total costs, especially when the costs of matching the remaining nodes and edges are small (F : maximum, ignores these costs). The variant with F : summation and f : Manhatan, performs slightly better than any other variant.

ARG matching with F : maximum, showed to be faster than the variants with F : summation, because of the faster pruning of the state space tree. However, searching with F : summation, is more accurate. In all experiments, ARG matching with F : maximum, exchanged at least a 10% loss in accuracy for 25% faster retrievals. Therefore, in searching an IDB with the ARG editing distance, accuracy and speed are traded-off.

Hungarian Method: Figure 10 illustrates the precision/recall diagram of the three variants of the Hungarian method. The variant with C : Manhatan, is the best achieving up to 10% better precision and better recall than the remaining two variants.

Hungarian Plus: Figure 11 illustrates the precision/recall diagram of the three variants of the Hungarian Plus method. The Hungarian method is used first to compute a mapping of objects between the

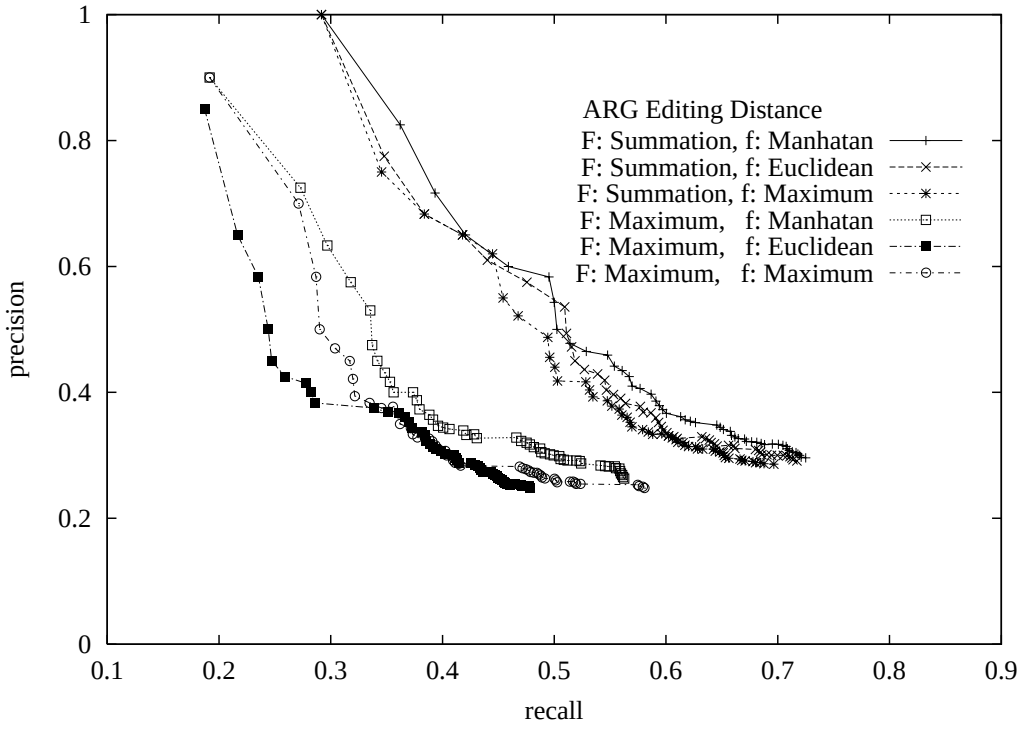


Figure 9: *Precision-Recall diagram of the ARG editing distance methods.*

images which are matched; Equation 8 is applied next to compute the cost of matching their relationships. The variant with C : Manhatan, is the best variant inheriting this behavior from its underlying Hungarian method which yielded a more accurate mapping than the remaining two variants. An interesting observation is that Hungarian Plus performs approximately the same with ARG matching with F: maximum.

2D Strings: Figure 12 illustrates the precision/recall diagram of the three 2D string variants corresponding to type 0, type 1 and type 2 matching respectively. Type 1 and type 2 matching are equivalent: The referee who performed the evaluations could not distinguish between their results. Both type 1 and type 2 matching perform better that type 0 achieving up to 15% percent better precision and better recall for all answers. The reason for this behavior lies on the fact that 2D string matching yields only binary (i.e., "yes/no") answers. The answers cannot be ranked and are listed in order of appearance (i.e., while the database is searched, each time a comparison evaluates to "yes" the matched image is appended in the answer set). Notice that, only the first 50 answers are evaluated. Any positive answers returned after the 50th image are missed. Type 0 matching missed a number of such answers (which evaluate to a positive judgment) while, type 1 and type 2 matching did not: Type 1 and type 2 matching is more selective than type 0 matching and their answers are subsets of the answers obtained by type 0 matching. These are the most promising answers, they are retrieved before others within the first 50 answers and are more likely to evaluate to a positive judgment.

An interesting observation is that, an evaluation based on answer sets with more than 50 (e.g., 100) answers would always indicate that type 0 matching is the most accurate method (type 1 and type 2

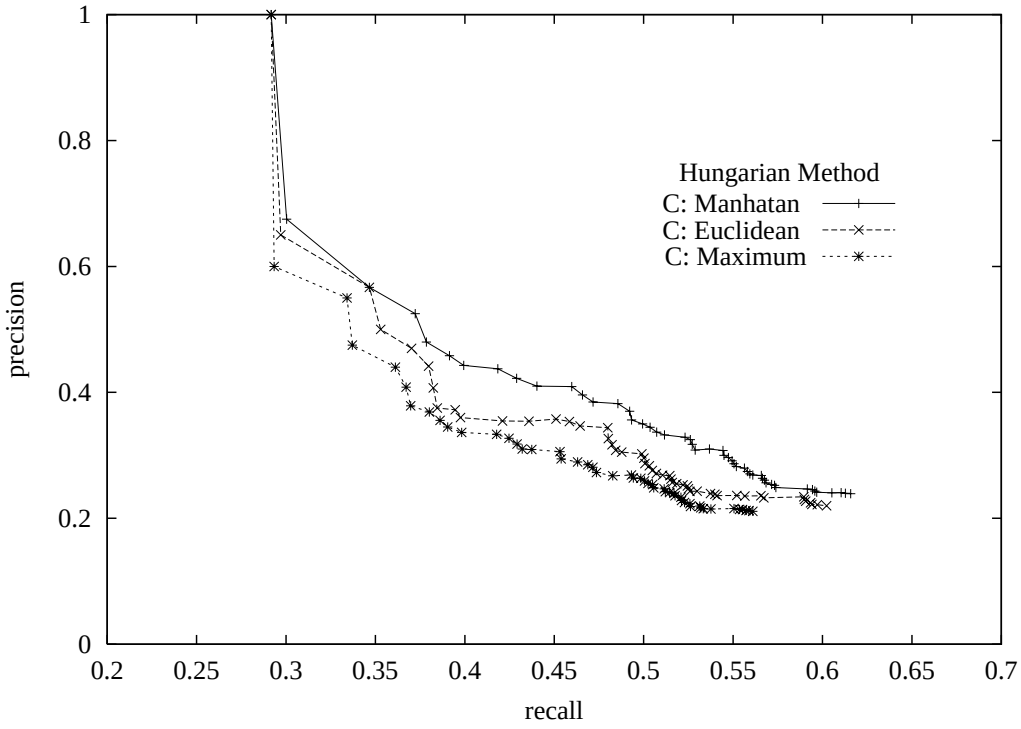


Figure 10: *Precision-Recall diagram of the Hungarian methods.*

matching are special cases of type 0 matching). Notice though that a typical user rarely retrieves more than 50 answers.

Expanded 2D Strings: Figure 13 illustrates the precision/recall diagram of the six variants of the expanded 2D string method. The $2DmatchA$ matching algorithm of [45] is applied. The variants corresponding to criterion 2 perform much better than those corresponding to criterion 1 achieving approximately 10% better precision and better recall for all answers. This is because matching with criterion 1, is more constraint than matching with criterion 2: With criterion 1, all objects have distinct ranks so that, it is less likely for objects in two images to have similar ranks.

The original 2D string methods discussed in the previous paragraph can be viewed as a special case of expanded 2D strings by applying criterion 2, by omitting the inclusion relationships from the representation and by keeping only one object property (i.e., size in this work). Therefore, the observations made before regarding performance apply here too except that, type 1 and type 2 matching perform only slightly better than type 0 matching: Type 0 matching misses less qualifying answers than in the previous case. This is expected because, type 0 matching utilizing more object properties and more relationships and is more selective than plain 2D string matching.

Type 0 and type 1 matching are equivalent for criterion 1: Compared to type 1, type 0 matching utilizes the additional condition that objects in the query image are allowed to have the same ranks on the horizontal or on the vertical axis, a condition that is never met with criterion 1 (i.e., criterion 1 doesn't allow for two or more objects to share the same position and have the same ranks). Therefore, type 0 and type 1 matching become equivalent for criterion 1. Retrievals with type 0 and type 1

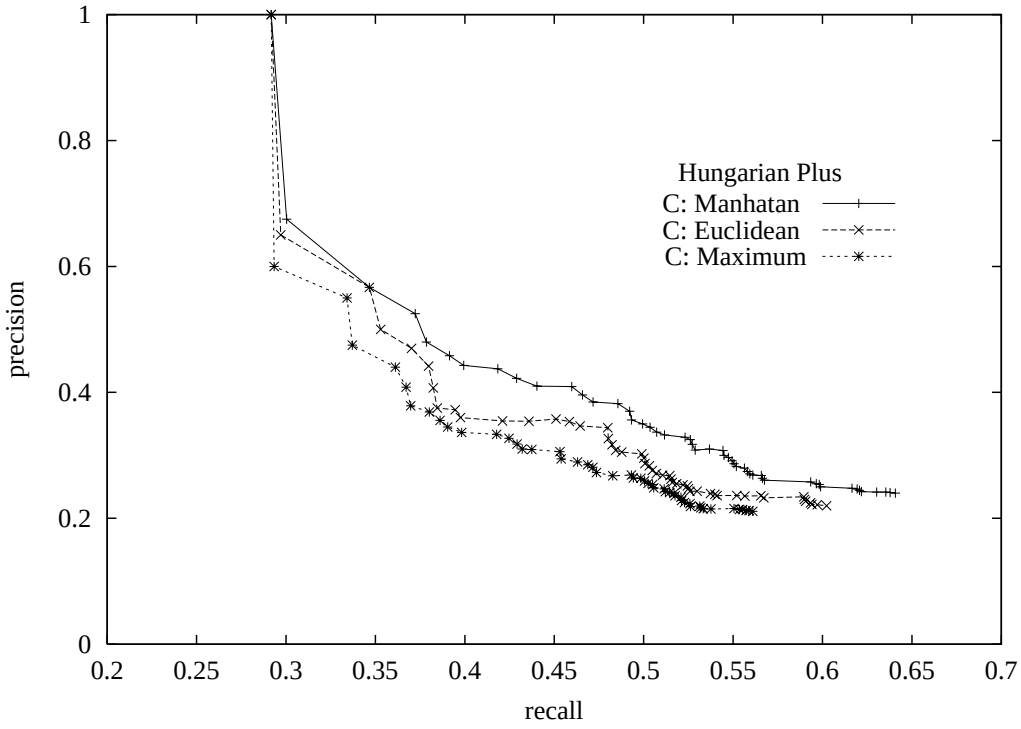


Figure 11: *Precision-Recall diagram of the Hungarian Plus methods.*

matching perform better than retrieval with type 2 matching. This is because type 2 matching is even more selective than type 0 and type 1 matching (by not allowing extra objects between the matched objects in a stored image) and misses answers which are retrieved by the other two criteria.

All Methods: Figure 14 illustrates the precision/recall diagram of the best variants of the five families of methods examined above which are

1. *ARG editing distance* with F : summation and f : Manhatan.
2. *Hungarian method* with C : Manhatan.
3. *Hungarian Plus* with C : Manhatan.
4. *2D strings* with type 2 matching.
5. *Expanded 2D strings* with criterion 2 and type 0 (or equivalently type 1) matching.

The ARG editing distance is obviously the best method achieving far better precision and recall than any other method for all answers, followed by Hungarian methods and 2D strings. An interesting observation is that that the accuracy of ARG matching can be improved further by expanding the ARG representation to include more node and arc attributes and relationships. An ARG may accommodate any number and kind of attributes that a domain expert may propose. This is not possible with the remaining methods which can handle only certain types of attributes and relationships.

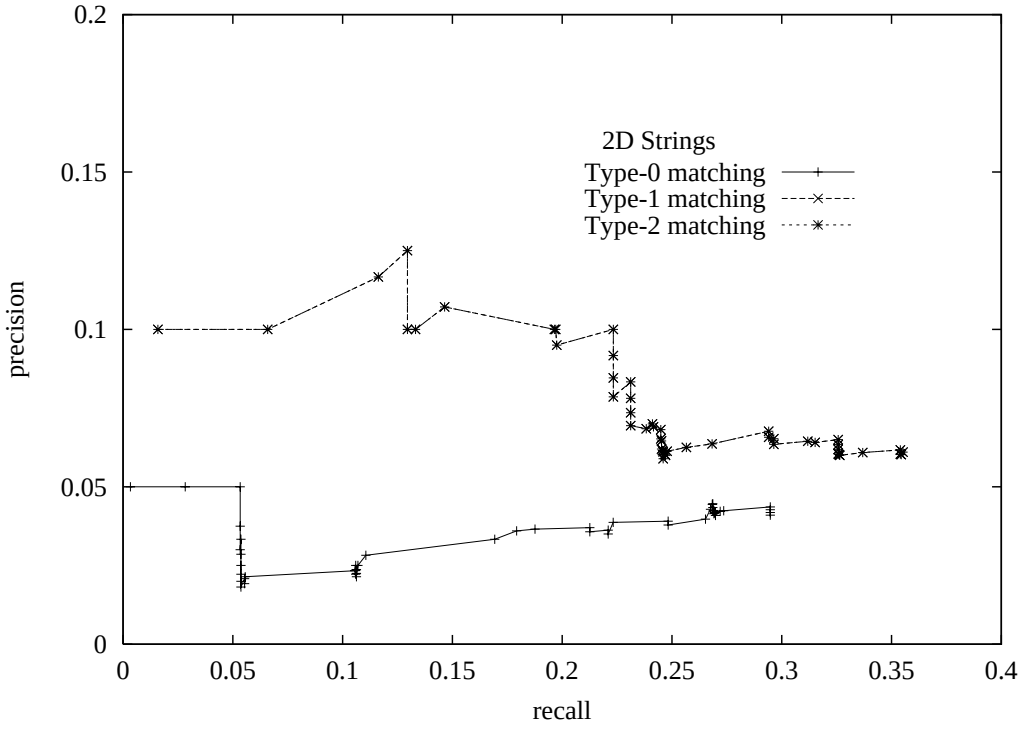


Figure 12: *Precision-Recall diagram of the original 2D string methods.*

Hungarian and Hungarian Plus are roughly equivalent with Hungarian Plus achieving slightly better recall. The Hungarian method always retrieves images with similar objects which, in most cases, happen to have the desired spatial relationships. Hungarian Plus takes the output of the Hungarian method in its input and, simply, rearranges the order of its answers by inspecting their spatial relationships using Equation 8. However, Hungarian Plus introduces only a few new images within the first 50 answers which are evaluated.

2D string methods are the worst, with expanded 2D strings achieving always better precision than the original 2D strings (i.e., matching is more selective by using more object properties and relationships and retrieves the most similar images first). However, for large answer sets with more than 35-40 images, 2D strings achieve better recall (i.e., matching with the expanded 2D strings is more constraint and may loose some qualifying images).

A final observation is that no method achieved recall greater than 75-80%. This means no method is 100% accurate; even the most accurate method (ARG matching) missed about 20% of the total number of correct answers (which are retrieved by other methods). A user might retrieve more correct answers by trying more than one methods with the same queries.

Ranking Quality: Table 2 shows the values of ranking quality (R_{norm}) computed to all methods. Based on R_{norm} alone, Hungarian Plus and Hungarian are the prevailing methods achieving $R_{norm} = 0.557$ followed by ARGs and 2D strings. This result contradicts the selection made based on precision and recall. R_{norm} measures the differences in the ranking of the results obtained by the referee and a method while, precision and recall measure the % percentages of qualifying images retrieved by

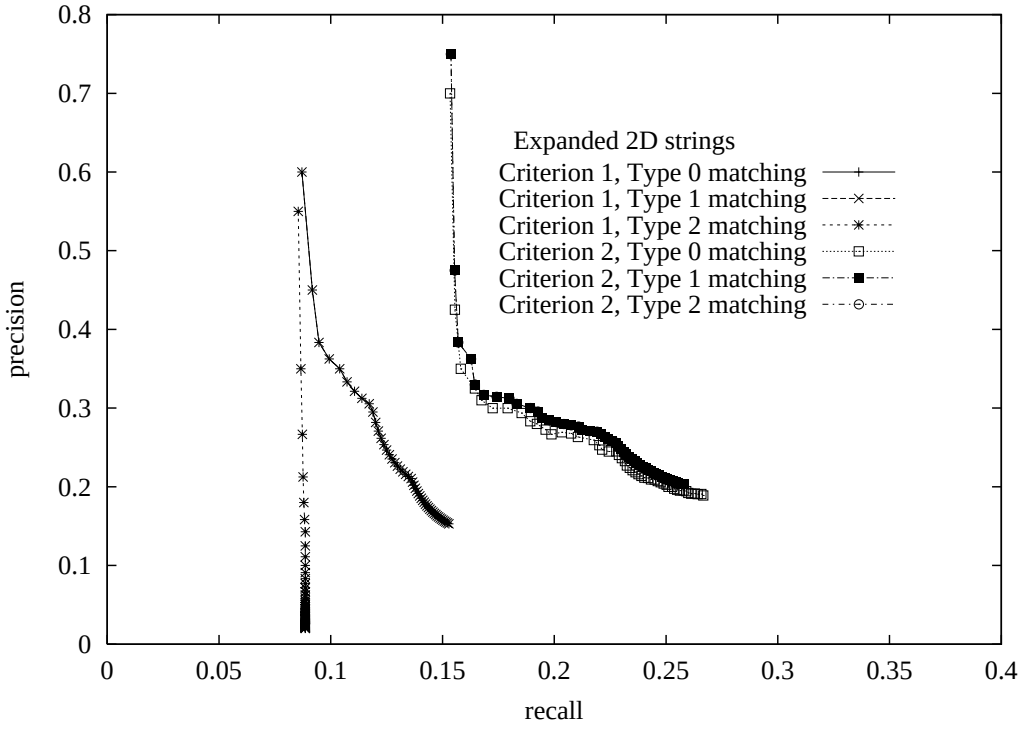


Figure 13: *Precision-Recall diagram of the expanded 2D string methods.*

a method. However, precision and recall are more characteristic measurements of the accuracy of a method and the selection of the most accurate method must be based on precision and recall rather than on ranking quality. However, ranking quality can be used in conjunction with precision and recall to select between equivalent methods.

2D strings, although less effective than ARG and Hungarian methods, produce high values of R_{norm}. This is because image matching with 2D strings is more selective so that, many of their first answers are correct. Notice that, 2D string matching yields binary answers, its answers cannot be ranked and their order in an answer set is random.

6.3 Discussion

The ARG editing distance, although the slowest, is the most accurate method. It is also the most general, that is, it can handle any number and type of object features and relationships. Hungarian and 2D string methods provide reduced complexity matching (i.e., polynomial) utilizing certain types of object features and relationships. Not all object features and all relationships can be handled by these methods. ARG and Hungarian methods, as opposed to 2D string methods which yield only binary ("yes"/"no") answers, allow for continuous measurements of the matching distance, and provide answers ranked by descending similarity with respect to the query.

The ARG editing distance is the slowest method because of its exponential time and space complexity of matching. Searching with this method can become even slower especially for ARGs with many nodes and many edges. One way of alleviating the complexity of matching would be to de-

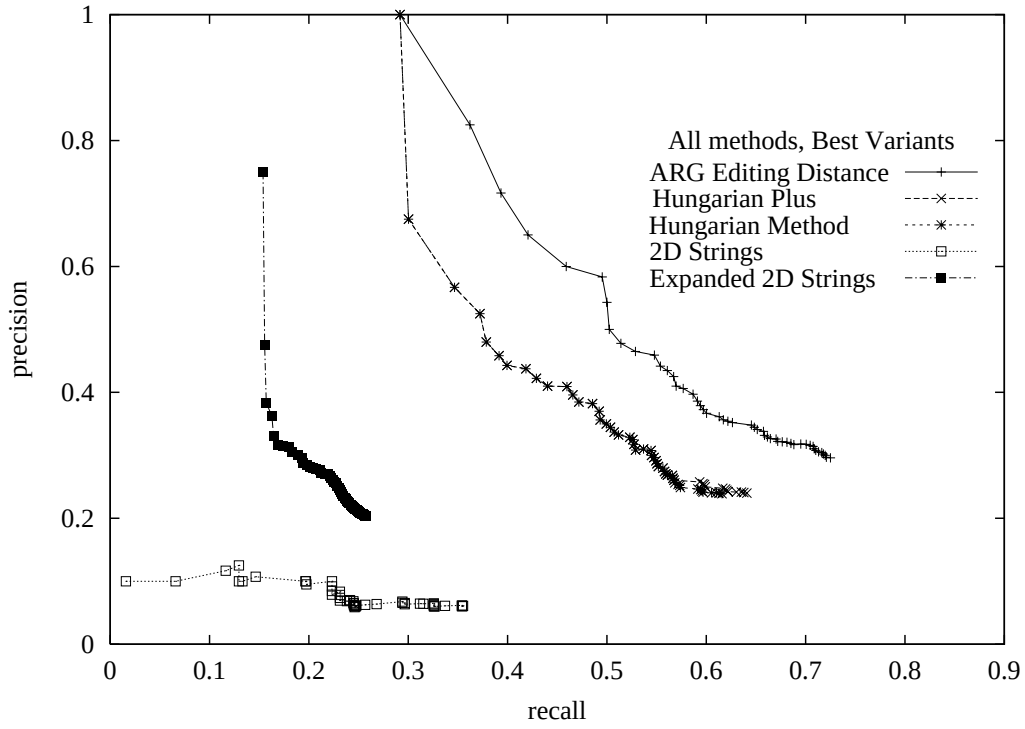


Figure 14: *Precision-Recall diagram of the best performance variants of each method family.*

sign ARGs with as few edges as possible by representing in ARGs only the most essential spatial relationships.

Hungarian Plus is expected to perform at least as good as the plain Hungarian method: It always introduces a correction on the results that the plain Hungarian method produces. 2D strings, in terms of accuracy, are expected to be always the worst methods because of their limited representation capabilities and the lack of ranking of their results. However, they are extremely fast and can be found especially useful in providing fast browsing of the database contents before a more accurate (but slower method is used). 2D strings can also be used in applications where one is only interested in retrieving some similar images for illustration purposes.

The analysis reveals that, in searching an IDB, accuracy and response time are traded-off: The most accurate a method the slower it is and the reverse. This is also demonstrated in Figure 15. The horizontal axis corresponds to the maximum recall achieved by the best variant of a method family and the vertical axis to its response time (in seconds).

A question arises as to whether the above results are general and applicable to more image types and datasets. Although safer decisions can be taken by applying the same evaluation on more datasets, the above discussion indicates that these results might be generally applicable to more image data types and (at least) for images with similar number of objects (i.e., 8-10). Notice also that the above analysis of performance is based on characteristics of the matching methods which are applied and not of the data used.

ARG Editing	Hungarian	Hungarian Plus	2D Strings	Expanded 2D Strings
F: Maximum f: Manhatan 0.346	C: Manhatan 0.086	C: Manhatan 0.089	2DmatchA type 0 0.425	criterion 1 0.002
F: Maximum f: Euclidean 0.351	C: Euclidean 0.557	C: Euclidean 0.557	2DmatchA type 1 0.395	criterion 2 0.049
F: Maximum f: Maximum 0.0749	C: Maximum 0.094	C: Maximum 0.0947	2DmatchA type 2 0.395	
F: Summation f: Manhatan 0.239			2Dquery1 type 1 0.395	
F: Summation f: Euclidean 0.529			2Dquery2 type 2 0.395	
F: Summation f: Maximum 0.0614				

Table 2: *Ranking quality R_{norm} computed to all methods.*

7 Conclusions

Several methods for answering spatial similarity queries are evaluated and compared. The evaluation is based on human relevance judgments following a well established methodology from the information retrieval field. A critical analysis of the performance of the method tested is presented. This analysis reveals that, in searching an image database by spatial image content, accuracy and retrieval response time are traded-off: The more accurate a method, the slower it is and the reverse. The experiments indicate that the ARG editing distance, although the slowest, has certain advantages over all other methods: (a) It is the most accurate demonstrating higher *precision* and *recall* than any other method, followed by Hungarian methods and 2D strings, (b) It is extendible: It can accommodate any type and number of properties of regions or relationships that a domain expert may appreciate. Future work includes experimentation with more datasets and methods and the investigation of indexing methods for speeding-up the retrieval response times of ARG matching methods.

Acknowledgments

I am grateful to Costas Georgiadis who implemented the ARG editing distance and the Hungarian methods for image matching. I am also grateful to Charalambos Genzis for his help in the experi-

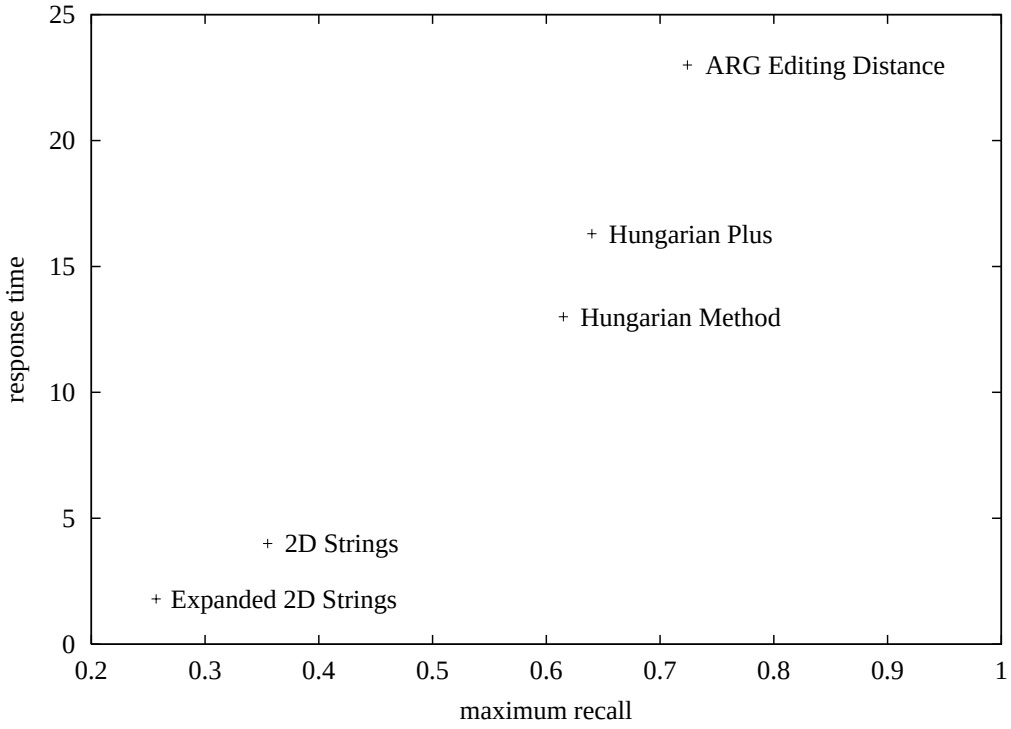


Figure 15: *Retrieval response time (in seconds) of the best performance variants of each method family plotted against recall.*

ments.

References

[1] 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).

[2] 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).

[3] 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).

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

[5] 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.

[6] 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).

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

[8] 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.

[9] 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.

[10] V. N. Gudivada and V. V. Raghavan. Modeling and Retrieving Images by Content. *Info. Proc. and Manag.*, 33(4):427–452, 1997.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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).

[16] E. Milios and E. G.M. Petrakis. Shape Retrieval Based on Dynamic Programming. *IEEE Trans. on Image Proc.*, 9(1):141–147, Jan. 2000. (http://www.ced.tuc.gr/~petrakis).

[17] Z. Rao, E. G.M. Petrakis, and E. Milios. Efficient Retrieval by Deformed and Occluded Shapes. In *Intern. Conference on Pattern Recognition (ICPR'00)*, volume 4, pages 67–71, Barcelona, Spain, September 2000.

[18] 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.

[19] 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.

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

[21] 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. (http://www.ced.tuc.gr/˜petrakis).

[22] 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.

[23] E. G.M. Petrakis and S. C. Orphanoudakis. A Generalized Approach to Image Indexing and Retrieval Based on 2-D Strings. In S.-K. Chang, E. Jungert, and G. Tortora, editors, *Intelligent Image Database Systems*, pages 197–218. World Scientific Pub. Co., 1996. (http://www.ced.tuc.gr/˜ petrakis).

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

[25] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, 1980.

[26] W.-H. Tsai and K.-S. Fu. Subgraph Error-Correcting Isomorphisms for Syntactic Pattern Recognition. *IEEE Trans. on Syst., Man, and Cybern.*, 13(1):48–62, 1983.

[27] 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.

[28] L. G. Shapiro and R. M. Haralick. Structural Discriptions and Inexact Matching. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 3(5):504–519, 1981.

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

[30] H. A. Almohamad and S. O. Duffuaa. A Linear Programming Approach for the Weighted Graph Matching Problem. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 5(15):522–525, May 1993.

[31] 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.

[32] K. E. Price. Relaxation Matching Techniques - A Comparison. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 7(5):617–623, Sept. 1985.

[33] H. S. Ranganath and L. J. Chipman. Fuzzy Relaxation Approach for Inexact Scene Matching. *Image and Vision Comp.*, 10(9):631–640, Jan. 1992.

[34] S. Gold and A. Rangarajan. A Graduated Assignment Algorithm for Graph Matching. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 18(4):522–525, April 1996.

[35] 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.

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

[37] L. G. Shapiro and R.M. Haralick. Organization of Relational Models for Scene Analysis. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 4(6):596–602, 1982.

[38] 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.

[39] 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.

[40] E. G.M. Petrakis, C. Faloutsos, and K.-Ip (David) 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).

[41] 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.

[42] 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.

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

[44] S. Berretti, A. Del Bimbo, and E. Vicario. Managing the Complexity of Match in Retrieval by Spatial Arrangement. In *Proc. of Intern. Conf. on Image Analysis and Proc. (ICIAP'99)*, Venice, Italy, Sept. 1999.

[45] E. G.M. Petrakis. *Image Representation, Indexing and Retrieval Based on Spatial Relationships and Properties of Objects*. PhD thesis, Univ. of Crete, Dept. of Comp. Science, March 1993. (http://www.ced.tuc.gr/~petrakis).

[46] E.M. Voorhees and D.K. Harmann, editors. *NIST Special Publication 500-242: The 7th Text REtrieval Conf. (TREC-7)*. Dept. of Commerce, Nat. Inst. of Standards and Technology, 1998. (http://trec.nist.gov/pubs/trec7/t7_proceedings.html).

[47] P. Bollmann and F. Jochum. Experimentelle Untersuchungen von Ranking-Verfahren. In *Deutscher Dokumentartag 1985*, K.G. Saur, Munich, New York, Paris, 1986.

Contents