INDEXING IMAGES WITH MULTIPLE REGIONS

Euripides G.M. Petrakis
Department of Electronic and Computer Engineering, Technical University of Crete
GR-73100 Chania, Crete, Greece
petrakis@ced.tuc.gr

Abstract. Similarity indexing using Spatial Access Methods (SAMs) like e.g., R-trees, assumes that each data entity (or query) is represented by exactly one multidimensional point. However, for several applications, including indexing and retrieval of multimedia data like one-dimensional signals and images, it is required that each data entity is represented by multiple points in a multidimensional space. This work extends the existing framework of indexing using SAMs to handle such data entities and has many desirable properties: For example, it provides index support for the two most common types of similarity queries, namely range and nearest neighbor (NN) queries, it returns exactly the same answers with the sequential scan method (only much faster) and, it works with any SAM and any data type. The effectiveness of the proposed approach is demonstrated using images with multiple regions.

1 Introduction

Traditionally, faster than sequential scanning in multimedia databases is achieved by means of the so called "Spatial Access Methods" (SAMs) like e.g., R-trees [2], SR-trees [4] etc. For example, methods such as [5] and [8], achieve fast retrieval in image databases by exploiting the assumption that each image corresponds to exactly one multidimensional point. SAMs cannot always treat multimedia data like one-dimensional signals and images which are represented by multiple multidimensional points. For example, [1] is a method for searching time-series databases to locate subsequences that match a query. The idea is to map each time sequence to multiple points in a multidimensional space which are indexed by an R-tree. In [7], the images are decomposed into subimages (each containing a fixed number of objects), the subimages are mapped to multidimensional points and are indexed using an R-tree. These methods treat only *range queries*: They retrieve entities within a known distance (tolerance) from the query by exploiting the assumption that the whole query matches a subset (or subsequence) of a stored entity. They do not treat *nearest neighbor* (or NN) queries (i.e., retrieve the k best matches). Each stored entity may be represented by multiple points but not the queries (each query is represented by exactly one multidimensional point).

This work focuses on image indexing by content and shows how a SAM can be used to accelerate the search for range and NN queries specifying one or more regions (or objects). The textbook approach to handle this information is Attributed Relational Graphs (ARGs). Each image (or query) ARG is represented by multiple points in a multidimensional space which are indexed by an R-tree. The performance of range and NN queries on ARGs is evaluated using the dataset of [7] and compared with the performance of sequential scanning. The results demonstrate very significant improvement in retrieval response times.

The emphasis of this work is not on choosing a good SAM, but on showing how an existing SAM (e.g., an R-tree) can be used to treat images and answer range and NN queries. Any SAM like e.g., SR-trees would do (in fact, a faster SAM would only make the method even faster). Two algorithms are proposed, one for type of query. 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 same algorithms can also be applied to time series, temporal,

spatio-temporal, DNA datasets etc.

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

2 Attributed Relational Graphs (ARGs)

It is assumed that each image has been segmented (automatically or manually) to more than one object or regions. Each object or region is represented by its surrounding contour. Fig. 1 shows an example of a segmented image (from the dataset of [7]) and its corresponding ARG. In this ARG, nodes correspond to regions and arcs correspond to relationships between regions. The arcs are directed from the outer to the contained region but their direction also depends on object labels: Arcs are 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.

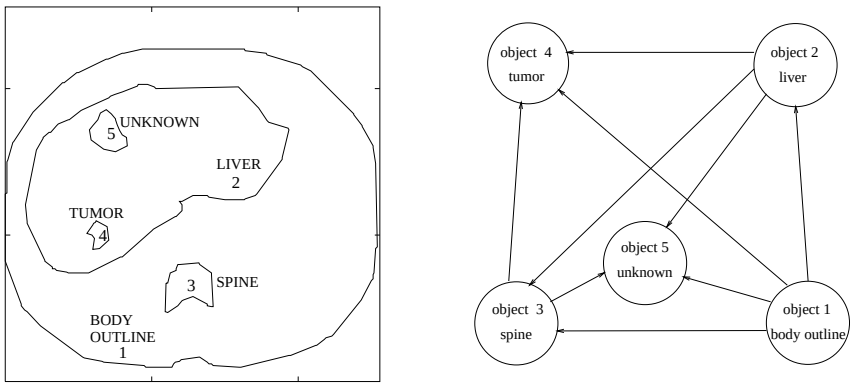

Figure 1: Example of a segmented image (left) and its ARG (right).

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 [7] 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 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 $[0, 2]$. 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 $(rd, 1 + rp, 1 + cos(ro), 1 + sin(ro))$.

Matching a query and a stored ARG is treated as an *"error-correcting subgraph isomorphism"* problem [6]: 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 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 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, $1 \leq i \leq l$. 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

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

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:

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

Notice that $\phi_1 \leq \phi_\infty$, $\phi_2 \leq \phi_\infty$ and $\phi \leq \Phi$, for any definition of ϕ and Φ respectively. There may exist extra nodes or edges in I 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 $S(I)$ of error transformations, but also finding the one with the minimum total cost. This can be formulated as a tree search problem which can be solved by an $A^\star$ algorithm [6].

3 Image indexing

Each image region is represented by its f-dimensionality feature vector, equivalently, by an f-dimensional point ($f = 5$ in this work). In turn, each image (or query) is represented by n points in the same f-dimensional space where n is the number of regions in the image (or query). Fig. 2 illustrates the mapping of three images (denoted as I,J, L) and one query (denoted as Q) in a two-dimensional space ($f = 2$). Image I consists of three regions (the I_1, I_2, I_3), J consists of exactly one region (the J_1) and, finally, image L consists of two regions (the L_1 and L_2). Query Q consists of two regions (the Q_1 and Q_2). The black circles denote the positions of the data

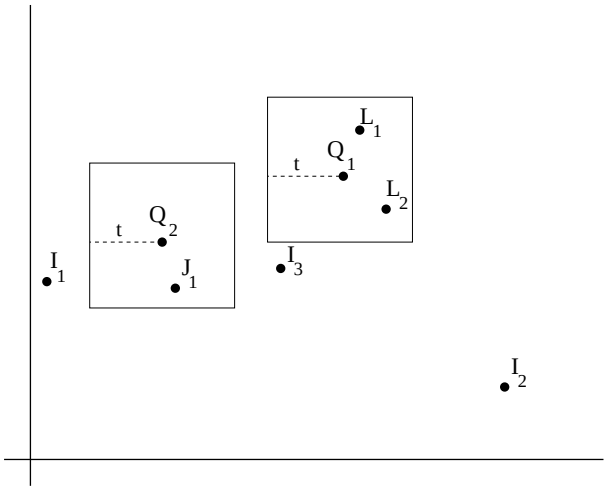

Figure 2: Mapping images and queries to points of a two-dimensional space.

and query regions in the above two-dimensional space. The rectangles centered around Q_1 and Q_2 denote range queries with tolerance t on each dimension (the query on Q_1 retrieves J_1 and, the query on Q_2 retrieves both L_1 and L_2).

More formally, a query Q is decomposed into n regions $Q_1, Q_2, \ldots Qn$; this is denoted as $Q = (Q_1, Q_2, \ldots, Q_n)$. Similarly, each stored image (model) I is decomposed into m regions $I_1, I_2, \ldots, I_m$; 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. Missing subscripts from Q_i or I_j denote regions 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 Fig. 2 $D_f = \phi_\infty$.

Actual distance $D(Q, I)$: It is the actual distance between Q and I. In this work, $D(Q, I)$ is the ARG editing distance between Q and I and can be defined either as an Φ_{sum} (summation of ϕ costs) or an Φ_{max} (maximum of ϕ costs) operation. 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. Existing search methods on SAMs (e.g., [2, 9, 3]) 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 be applied directly for answering range and NN queries specifying multiple regions.

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

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

1. Decompose Q into $Q_1, Q_2, \ldots Q_n$ regions (f-dimensional points).
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 3: Algorithm to retrieve all images within distance t from a query using a SAM.

Definition 1 (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 definition of ARG distance of Section 2 satisfies the lower bounding principle.*

Proof 1 *D is the total ARG editing distance and it is defined by Eq. 2 or by Eq. 3. D_f corresponds to the cost of an editing operation and is defined by Eq. 4 or by Eq. 5. It is easy to see that the cost of an edit operation is always less than the total editing distance D.* □

3.1 Range queries

The basic idea in processing range queries is to search the index for each Q_i using D_f and to combine their results. Fig. 3 illustrates an algorithm to retrieve all images I within distance t from Q.

Lemma 2 *The algorithm of Fig. 3 guarantees no false dismissals.*

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.* □.

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 at step 4. The performance of the algorithm increases exponentially with the tolerance (i.e., the number of points around the query increases with t^f). Therefore, the lower the t, the faster the algorithm is.

3.2 Nearest Neighbor (NN) queries

Algorithms for NN searching on SAMs like e.g., [9, 3] 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 $D(Q, I)$. Fig. 4 illustrates an algorithm to retrieve the k most similar images of Q using a SAM.

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 $Q_1, Q_2, \ldots Q_n$ regions (f-dimensional points).
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 Fig. 3.
5. Output the k images closest to Q.

Figure 4: Algorithm to retrieve the k NNs of a query with regard to distance D, using a SAM.

Lemma 3 *The algorithm of Fig. 4 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 Fig. 4 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 (i.e., k distinct image identifiers I). A conventional NN algorithm (e.g., [9]) would return the k NNs 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 [3] so that the extra NNs are obtained without having to recompute the first k neighbors. In this work, such an incremental algorithm was not available. The results reported in Section 4 are obtained using the NN algorithm of the DR-tree (a version of the R-tree developed at the Univ. of Maryland), a variant of the algorithm of [9]. The experiments showed that even this non-incremental algorithm retrieved the actual k-NNs with almost 100% accuracy.

The performance of the algorithm depends on the value of t applied at step 4. The lower the value of t the faster the algorithm is. Low values of t can be achieved by designing the distance D to take values much less than the range of values of each indexed attribute ([0,2] in this work).

4 Experiments

The method is implemented in C and the experiments were run on a dedicated SUN Ultra 1 (167MHz) with 128Mbytes main memory running Solaris^{TM}. The dataset of $13,500$ segmented

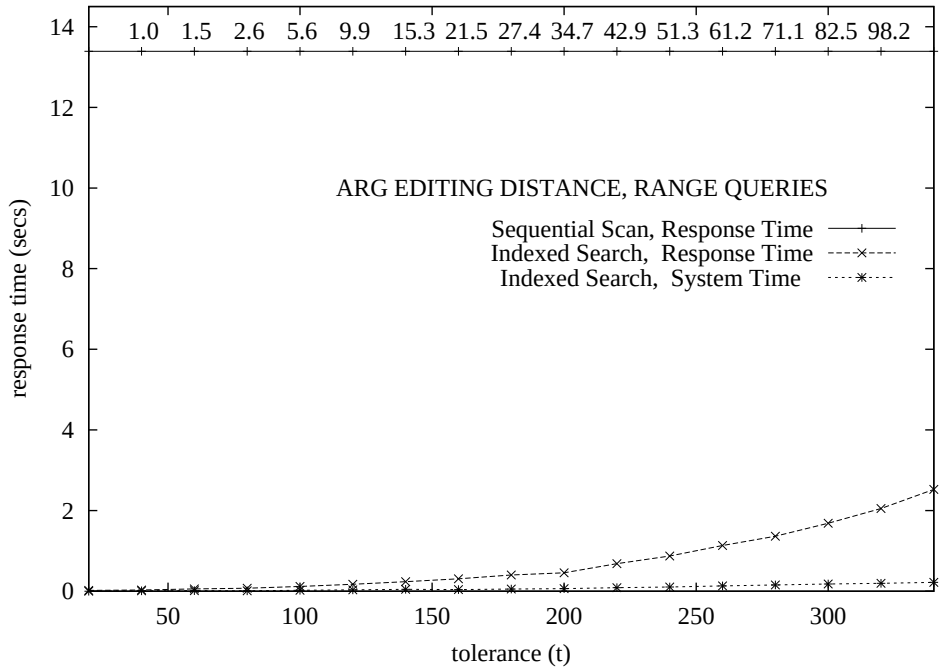

Figure 5: Average retrieval response times as a function of the tolerance t corresponding to sequential and indexed search.

images of [7] was used as a testbed. All images contain between 4 and 8 objects. In all the experiments $f = 5$. The R-tree 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 23.8 Mbytes. 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. The queries contain between 4 and 6 objects. The queries are applied (one after the other) on the same R-tree and all results are averages over 20 queries. The experiments are designed to demonstrate the superiority of the proposed approach for range and NN searching 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.

In all the experiments below Φ_{max} and ϕ_1 are used. The reason for this selection has to do also with the efficient processing of NN queries: The NN algorithm call for range queries with tolerance t at step 4; the smaller the value of t the faster the algorithm is. First, Φ_{max} guarantees that D takes values in the range $[0, 2]$ (a range query with tolerance larger than 2 would retrieve the entire R-tree!). Second, $\phi_1 \leq \phi_\infty$ so that, the combination of Φ_{max} and ϕ_1 guarantees smaller values of D than Φ_{max} and ϕ_∞ and therefore, smaller values of the tolerance t (ϕ_2 can also be used since $\phi_2 \leq \phi_\infty$).

Fig. 5 plots the response time for range queries as a function of the tolerance t for both indexed and sequential search. The labels above the sequential method denote average numbers of retrieved images. The proposed method achieves large savings, even for high values of the tolerance (e.g., for $t = 200$ the retrieved set is almost 35 images). Even for such large queries the response times are below 2 seconds. Notice that most of the response time is cpu (user) time devoted mainly to ARG distance calculations (clean-up). Only a small percentage is system time spent for R-tree and ARG file search.

For the proposed method, the shape of the curve resembles an exponential curve. This is expected because the number of candidate regions which are retrieved by D_f increases exponentially with the tolerance (i.e., with t^f). The more the retrieved regions the slower the retrieval

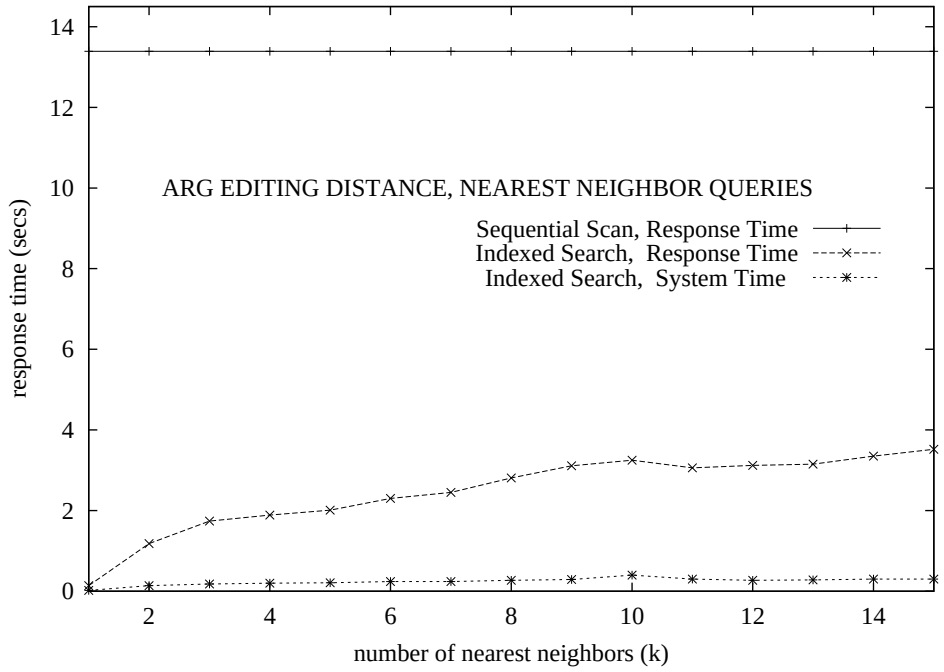

Figure 6: Average retrieval response times as a function of the number of nearest neighbors (NNs) retrieved corresponding to sequential and indexed search.

response times are (all these regions account for ARG distance calculations in clean-up).

Fig. 6 (right) plots the response time as a function of k, the number of retrieves NNs. Notice that, response time increases only for $k < 10$ and remains practically constant for greater values of k. Retrieval response time depends on the tolerance t which is computed at steps 2-3 of the algorithm of Fig. 4. The value computed for small k seems to be also adequate for retrieving even more NNs. Again, system time is only as small percentage of the total response time.

5 Conclusions

This work shows how a spatial access method can be used to answer range and nearest neighbor queries specifying multiple regions. Two algorithms are proposed, one for each type of query. Both 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. The emphasis is not on a particular image type or on choosing a good set of features for this image type, but on speeding-up the retrieval response times for a given set of features that a domain expert has proposed. The method can treat any set of features and any image type. The results demonstrate significant performance improvements compared to sequential scanning.

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 ARG matching algorithm.

References

[1] Faloutsos C, Ranganathan M, and Manolopoulos Y (1994) Fast Subsequence Matching in Time-Series Databases. Proc. of ACM SIGMOD, pages 419–429.

[2] Guttman A (1984) R-trees: A Dynamic Index Structure for Spatial Searching. Proc. of ACM SIGMOD, pages 47–57.

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

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

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

[6] Messmer BT (1995) Efficient Graph Matching Algorithms. PhD thesis, Univ. of Bern, Switzerland. http://iamwww.unibe.ch/~fkiwww/projects/GraphMatch.html.

[7] Petrakis EGM and Faloutsos C (1997) Similarity Searching in Medical Image Databases. IEEE Trans. on Knowl. and Data Eng., 9(3):435–447.

[8] Petrakis EGM, Faloutsos C and Lin KI (to appear) ImageMap: An Image Indexing Method Based on Spatial Similarity. IEEE Trans. on Knowl. and Data Eng. http://www.ced.tuc.gr/~petrakis.

[9] Roussopoulos N, Kelley S, and Vincent F (1995) Nearest Neighbor Queries. Proc. of ACM-SIGMOD, pages 71–79.