Similarity based Retrieval of Pictures Using Indices on Spatial Relationships

A. Prasad Sistla Clement Yu Chengwen Liu King Liu

Department of Electrical Engineering and Computer Science,
University of Illinois at Chicago,
Chicago, Illinois 60680
sistla@surya.eecs.uic.edu, yu@dbis.eecs.uic.edu

Abstract

The paper presents (i) two similarity based methods for retrieval of pictures using indices on spatial relationships; (ii) efficient algorithms for the deduction and reduction of spatial relationships, which are used for computation of similarity values; (iii) identification of a desirable property for spatial similarity functions and the construction of such a function.

1 Introduction

We are currently witnessing an explosion of interest in multimedia technology. Consequently, pictorial and video databases will become central components of many future applications. Access to such databases will be facilitated by a query processing mechanism that retrieves pictures based on user queries. Existing pictorial database management systems have been mostly application dependent (for example see [Amd93, LeeW93, RP92, Ch92]). Some preliminary work towards a unified framework for content based retrieval of images can be found in [GWJ91, CK81, Car93]. Our motivation is to construct a set of basic tools that would be applicable in pictorial databases for a broad *class* of applications. These tools consist of components to reason about spatial relationships, to handle user interfaces, and to compute degrees of similarity between queries and pictures etc. In this paper, we concentrate on methods for similarity based retrieval of pictures using indices on spatial relationships.

We assume that there is a database containing the pictures. We also assume that each picture is associated with some meta-data describing the contents of the picture. This meta-data contains information about the objects in the picture, their properties and the relationships among them. For example, consider a picture containing a tall man shaking hands with a slender woman and is to the left of the woman. The meta-data about this picture identifies two objects,a man and a woman with attribute values "tall" and "slender" respectively, and the spatial relationship *left_of* and the non-spatial relationship hand-shaking. We assume that this meta-data is generated a priori (possibly, by image analysis algorithms, or manually, or by a combination of both), and is stored in a separate database. This meta-data will be used by the query processing mechanism in determining the pictures that need to be retrieved in response to a query. The meta-data facilitates efficient query processing, i.e. it avoids the invocation of the expensive image analysis algorithms each time a query is processed.

Similarity based retrieval of pictures consists of computing a similarity value with each picture that denotes how closely the picture matches the query, and retrieving those pictures with the highest similarity values. Such retrievals are needed when the user cannot provide a precise specification of what he/she wants. Even if the user can precisely specify his/her requirements, there may not be any pictures in the database that exactly match with the user's query, and in this case the user may want the closest matches. It is to be noted that, when we use similarity based retrieval, any picture that matches exactly with the query will be assigned the maximum similarity value,

Proceedings of the 21st VLDB Conference
Zurich, Switzerland, 1995

and will automatically be retrieved first.

In this paper, we consider similarity based picture retrieval using various spatial relationships that are of general interest. Specifically, the following relationships— *left_of*, *right_of*, *in_front_of*, *behind*, *above*, *below*, *inside*, *outside* and *overlaps* — are used.

When computing the similarity value of a picture, in general, we need to check if the picture satisfies any of the spatial relationships which are implied by the user's query, in addition to the explicitly stated relationships. For this reason, it becomes important to compute all the implied spatial relationships of the query. On the other hand, some of the spatial relationships of the user's query which are satisfied by the picture may be redundant, i.e. they may be implied by other satisfied relationships; such redundant relationships should be given lower weights or they should not be considered in the computation of the similarity value of the picture. For this reason, it becomes essential to compute the reduction of a set F of spatial relationships which is the smallest subset of F that implies all the relationships in F.

In this paper, we present efficient algorithms for the *deduction* problem, i.e. the problem of deducing all the implied relationships of a given set of relationships, and the *reduction* problem, i.e. the problem of computing the reduction of a given set of relationships. The deduction algorithm is based on a complete set of rules for deducing spatial relationships; such rules were presented in our earlier paper [SYH94]; one way of deducing the implied relationships is to apply a general purpose deductive database system such as LDL to such rules; however, such a method may be less efficient than a direct algorithm. For this reason, we develop a direct algorithm for the deduction problem. The deduction problem is more general than the transitive closure problem. However, our deduction algorithm has the same complexity as that of the most efficient known algorithm for the transitive closure problem. Both of our deduction and the reduction algorithms employ an automata based approach and have the same time complexity. Both these algorithms are used in computations of similarities for retrieval of pictures.

Given a picture and a query, the problem of computing the similarity value of the picture can be shown to be NP-hard, when this value is given by any *sound* similarity function; a similarity function is *sound* if it assigns strictly highest value to pictures having an exact match, i.e. they satisfy all the conditions of the query. The high complexity of computing a similarity value given by a sound similarity function is due to the fact that, each object in the query may match with multiple objects in the picture, and if there are many objects in the query then the number of combinations of matchings of different objects may grow exponentially.

To overcome the above problem, we *restrict* the matchings so that each object in the query is matched with the most similar object in the picture based on the attribute values of the corresponding objects (if there are multiple most similar objects then one of them is chosen based on other criteria), and define our similarity function with respect to this *maximal* matching; we call the resulting similarity values as *restricted* similarity values. Earlier experimental results of [ATY95], where deduction and reductions have not been employed, confirm this to be a good strategy. The restricted similarity functions, that we use, can be computed efficiently; clearly, they are not sound in the general case. However, they can be shown to be sound when the picture has exactly one maximal matching with respect to the query. Using the maximal matching, a restricted similarity value is computed as the sum of three other similarity values; the first two are based on matching of the objects and non-spatial relationships respectively; the last one is based on the spatial relationships and is given by a *spatial similarity* function.

We identify an important desirable property of spatial similarity functions called *monotonicity*; intuitively, monotonicity requires that a picture satisfying more spatial relationships of the query than another picture should be given a higher spatial similarity value. We construct a similarity function having such a property.

When the number of pictures in the database is very large, it becomes impractical to explicitly examine each picture to compute its spatial similarity value. We avoid this problem by employing indices on the spatial relationships. We present two efficient methods for computing the similarity values of those pictures that satisfy at least one of the spatial relationships in the query. Furthermore, these methods employ the deduction and reduction algorithms in the computation of the similarity values. We suppose that the user specifies an integer u and would like to retrieve the u pictures having the highest similarity values. The first method computes the exact spatial similarity values and retrieves the u pictures having the highest spatial similarity values; it is very efficient when the number of spatial relationships given in the query is small.

The second method estimates the spatial similarity values, and retrieves the meta data of the cu pictures having the highest estimated spatial similarity values for some constant $c > 1$; it then computes the actual spatial similarity values of these and retrieves the u pictures, among these cu pictures, having the highest spatial similarity values; this method is efficient if the number of relationships is large. Experimental results

for the second method show that for a value of $c = 3$, in all practical cases, the u pictures with the highest spatial similarity values among all the pictures in the database are contained in the cu pictures having the highest estimated spatial similarity values, and will therefore be retrieved.

Although the above methods presented in this paper compute the similarity values using the maximal matching, they can be modified easily to consider all possible matchings. However, this will cause an increase in complexity of the resulting methods due to the NP-hardness result.

In summary, the following are the major contributions of this paper.

- An efficient algorithm for the deduction problem;
- An efficient algorithm for the reduction problem;
- Identification of the monotonicity property for spatial similarity functions and the construction of such a function.
- Two efficient methods for computation of spatial similarity values that employ indices on spatial relationships and that employ the deduction and reduction algorithms.

There has been some earlier work [CSY84, CCT94, GR94, RM89] on handling spatial relationships. These works consider exact match for picture retrieval, as opposed to our similarity based approach. Also, they use a different set of spatial operators, and do not employ indices. More recent works [GZCS94, Ni93, HOP91, LSY89] consider similarity based retrieval of pictures using indices. However, their retrieval is primarily based on color variations, shapes of objects and texture. They use low level meta-data whereas our meta-data is at a higher level. They employ neither deduction nor reduction for handling the spatial relationships. There has also been much work on handling spatial relationships in GIS (for example, see [Eg89]). None of these works uses similarity based retrieval using indices and employing deduction/reduction. Surveys of pictorial database systems can be found in [TaY84, GrM92, ChH92].

This paper is organized as follows. Section 2 presents the notation and various definitions used in the remainder of the paper. Section 3 gives the algorithm for the deduction problem. Section 4 presents the algorithm for computing the minimal reduction. Section 5 presents some desirable properties of spatial similarity functions and presents a method for obtaining such similarity functions. Section 6 presents two methods for computing similarity values using indices and gives some preliminary experimental results. Section 7 contains conclusions and discusses future work.

2 Notation and Definitions

We assume that each object has a unique name associated with it belonging to a finite set of names N. Each 3-dimensional picture specifies a set of points occupied by each object present in the picture. Formally, a picture p is a mapping that maps each object A to a set of points $p(A)$ in the 3-dimensional space.

We consider the following set of spatial relationship symbols— *left_of*, *behind*, *above*, *below*, *inside*, *outside*, and *overlaps*.

Let p be a picture in which objects A and B are present. Now, we formally define when p satisfies the above relationships. p satisfies the relationship A *left_of* B iff the x-coordinate of every point in $p(A)$ is less than the x-coordinate of every point in $p(B)$. Similarly, semantics of the *above* and *behind* relationships are defined using the y and z-coordinates, respectively. p satisfies A *inside* B iff $p(A) \subseteq p(B)$. p satisfies A *outside* B iff $p(A) \cap p(B) = \emptyset$. p satisfies A *overlaps* B iff $p(A) \cap p(B) \neq \emptyset$.

Let F be a finite set of relationships. We say that F is consistent if there exists a picture that satisfies all the relationships in F. We say that a relationship r is *implied by* F, if every picture that satisfies all the relationships in F, also satisfies the relationship r. For example, the set of relationships $\{A\ left\_of\ B\ ,\ B\ left\_of\ C\}$ implies $A\ left\_of\ C$.

3 Deduction Algorithm

Appendix A presents a set of rules (taken from [SYH94]) for deducing new relationships from a given set F of relationships. Rule I captures the transitivity of the relationships *left_of*, *above*, *behind* and *inside*. Rule II denotes the interaction between *left_of*, *above*, *behind* relationships with the *overlaps* relationship. Rule III captures the interaction of the relationships *left_of*, *above*, *behind*, *outside* with the *inside* relationship.

For the set F, let $ded(F)$ denote the set of all relationships deducible from F using the rules of [SYH94]. By the soundness and completeness result given in [SYH94], $ded(F)$ is identical to the set of relationships implied by F. In this section, we present an efficient algorithm, based on the rules, for directly computing $ded(F)$. Let n and m be the number of objects and the number of relationships in F. All the *inside* relationships in $ded(F)$ are computed by simply taking the transitive closure of the *inside* relationships in F. All the *overlaps* and *outside* relationships in $ded(F)$ are easily computed.

The algorithms for the relationships *left_of*, *above* and *behind* are more complicated. We will consider the *above* relationship here. The other two relationships can be similarly handled. In this algorithm, we use

a derived relationship *contains*, which is the dual of *inside*, defined as follows. *A contains B* is true iff *B inside A* is true. Intuitively, *A contains B* is satisfied if object *B* is inside *A*.

Our algorithm for deducing the *above* relationships is based on using a simple finite state automaton $\mathcal{A}$ which is shown in figure 1 given below.

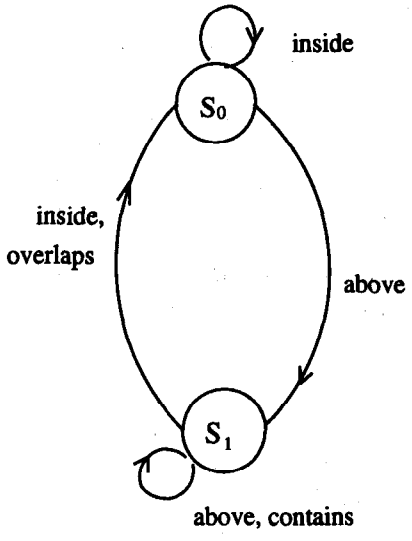

Figure 1: Automaton used for deduction

The automaton $\mathcal{A}$ has two states s_0 and s_1. The states s_0 and s_1 are the initial and final states respectively. In order to explain the significance of the automaton, we need the following definition. A *chain* of relationships is a sequence of relationships in which the first object of each succeeding relationship is the same as the second object of its preceding relationship. For example, A_1 *inside* A_2, A_2 *inside* A_3, A_3 *above* A_4 is a chain. Intuitively, the automaton captures a chain of relationships that can be used to deduce an *above* relationship. For example, an *above* relationship can be deduced by a chain of *inside* relationships followed by one or more *above* relationships. More precisely, *A above B* can be deduced from the chain *A inside* B_1, B_1 *inside* B_2, B_2 *above B*. In this deduction, we first use transitivity of *inside* (rule I) to deduce *A inside* B_2 and use rule IIIa to deduce *A above B*. Note that the transitivity of *inside* is captured by the self loop in state s_0. Rule IIIa is captured by the self loop in state s_0 and by the transition from s_0 to s_1. Rule IIIb is captured by the transition from s_0 to s_1 and the self loops in state s_1 (Note that *B contains C* is the same as *C inside B*). The transitivity of *above* is captured by the transition from s_0 to s_1 and the self loop in state s_1. Similarly, *A above B* can be deduced by a chain consisting of a sequence one or more *above* relationships followed by a single *overlaps* relationship, followed by one or more *above* relationships. This interaction between *overlaps* and *above*, given by rule II, is captured by the self loops in state s_1, by the transition from s_1 to s_0 labeled with *overlaps*, and the transitions from s_0 to s_1 labeled with *above*.

Now, we define a labeled graph $\mathcal{G}$ which captures all the *inside, contains, above* and *overlaps* relationships in F. These are the only relationship symbols relevant in the deduction of *above* relationships. The nodes of the graph are objects. The edges of $\mathcal{G}$ are labeled with relationship symbols. For distinct objects A and B, there exists an edge from A to B labeled with the relationship symbol $x \in \{inside, above, overlaps\}$ if $A\ x\ B$ is a relationship in F. Also, there exists an edge from A to B labeled with *contains* if *B inside A* is a relationship in F. Note that any path in $\mathcal{G}$ represents a chain of relationships. The number of nodes in $\mathcal{G}$ is n and the number of edges is $O(m)$.

From the automaton $\mathcal{A}$ and the labeled graph $\mathcal{G}$, we define another labeled graph $\mathcal{C}$ as follows. Intuitively, $\mathcal{C}$ denotes the simulation of the automaton $\mathcal{A}$ on all paths contained in $\mathcal{G}$. For each object A and state s_i of the automaton $\mathcal{A}$, there exists a single node (A, s_i) in $\mathcal{C}$, and these are the only nodes in $\mathcal{C}$. There exists an edge in $\mathcal{C}$ from (A, s_i) to (B, s_j) labeled with the relationship symbol x, if there exists an edge in $\mathcal{G}$ from A to B that is labeled with x, and there is a transition of the automaton $\mathcal{A}$ from state s_i to s_j on input x. These are the only edges in $\mathcal{C}$.

THOEREM 3.1: There exists a path in $\mathcal{C}$ from the node (A, s_0) to the node (B, s_1) iff the relationship *A above B* is in $ded(F)$.

Figure 2 given below shows the graphs $\mathcal{G}$ and $\mathcal{C}$ for the case when F is the set of relationships { *A above B, A above C, C inside B, C above D* }. There is a path from (A, s_0) to (D, s_1) in $\mathcal{C}$. By theorem 3.1, it follows that *A above D* is deducible from F.

We use the graph $\mathcal{C}$ in the algorithms for computing the set of all *above* relationships deducible from F. Note that the number of nodes in $\mathcal{C}$ is $2n$ and the number of edges is $O(m)$. To compute the set of all *above* relationships deducible from F, we compute the transitive closure of $\mathcal{C}$, and output the relationship *A above B* for each edge from a node of the form (A, s_0) to a node of the form (B, s_1) in the transitive closure. ¿From standard graph algorithms, the transitive closure of $\mathcal{C}$ can be computed in time $O(nm)$. Hence the complexity of this algorithm is $O(nm)$.

4 Reduction Algorithm

In this section, we consider the problem of computing a *minimal reduction* G of a set of relationships F,i.e. a minimal set of relationships G contained in F such that every relationship deducible from F is also deducible from G. In general, minimal reductions of F may not

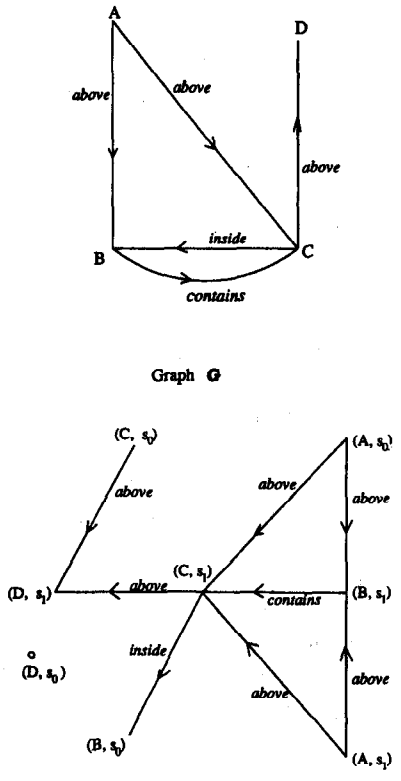

Figure 2:

be unique due to the following reason. Suppose the relationship A *overlaps* B is in a minimal reduction. We can get another minimal reduction by replacing A *overlaps* B by its dual relationship B *overlaps* A (due to the symmetry of *overlaps*). We avoid this problem by considering the *overlaps* relationship as unordered pair, not as an ordered pair. Similarly, we identify dual *outside* relationships. Furthermore, we also make the assumption (*)

(*) for any two distinct objects A and B it is not possible to deduce both the relationships A *inside* B and B *inside* A.

Under the above assumptions, the minimal reduction of F is unique as indicated by theorem 4.1. We let *red*(F) denote this minimal reduction.

THEOREM 4.1: The minimal reduction of F is unique.

Now, we show that the set of *above* relationships in a minimal reduction of F is unique, and show how these relationships can be efficiently computed. We will use the graph C defined in the previous subsection for this purpose. It can be shown that the graph C is acyclic.

Now, we define a simple relation, called *uses*, among the *above* relationships in F. We say that an *above* relationship $r \in F$ *uses* another *above* relationship $r' \in F$, if there exists a deduction of r from the relationships in $F - \{r\}$ such that one of the steps in the deduction employs r'. Intuitively, the *uses* tells us that, if all the relationships that r uses are present in the reduction of F, then r need not be present in the reduction. We say that an *above* relationship r is *required* if there is no other *above* relationship r' such that r uses r'. The following lemma shows that the "uses" relation is acyclic. It relates the the required relationships with the minimal reduction of F.

LEMMA 4.2: The relation *uses* is acyclic. Further more, the set of *above* relationships in a minimal reduction of F is exactly the set of required relationships.

Let A *above* B, denoted by r, be any *above* relationship on F. Using theorem 3.1, it is easy to see that r is deducible from the other relationships in F iff there is a path in C of length greater than one from the node (A, s_0) to the node (B, s_1). Hence, r is a required relationship, iff the only path from the node (A, s_0) to (B, s_1) is the *above* edge joining these two nodes, iff the length of the longest path from the node (A, s_0) to the node (B, s_1) is one. Now, to compute the set of required relationships in F, we simply compute the length of the longest paths from every pair of nodes in C , and identify edges satisfying the previous property. The lengths of the longest paths between every pair of vertices in an acyclic graph can be computed in time $O(ve)$ where v, e are the number of vertices and edges in the graph respectively. In our case, $v = O(n)$ and $e = O(m)$. Hence, by lemma 4.2, the set of above relationships in the minimal reduction of F can be computed in time $O(nm)$.

Thus, we see that the minimal reduction of F is unique and can be computed in time $O(nm)$.

For the example given by figure 2, we see that the relationships A *above* B, C *above* D are in the minimal reduction since the lengths of the longest paths from (A, s_0) to (B, s_1) and from (C, s_0) to (D, s_1) are both equal to 1. It is also to be noted that A *above* C is not in the reduction since the length of the longest path from (A, s_0) to (C, s_1) is 2.

5 Similarity Based Retrieval

In this section, we consider similarity based retrieval systems. A similarity function is a function that associates a real number with every pair of a query and a picture; intuitively, the value given by the function denotes how closely the picture satisfies the query, the higher the value the closer the picture is to the query.

Let Q be a query and P be a picture. A matching ρ is a partial function that maps each object in Q to a distinct object in P. We say that a picture P is an *exact match* of a query Q if there exists a total matching ρ so that each object in Q is mapped to an object of the same type and having the same attribute values, and such that all relationships among objects

in Q are satisfied by the corresponding objects in P (specified by ρ). We say that a similarity function f is *sound* if for each query Q, and for every two pictures P_1 and P_2 such that P_1 is an exact match of Q and P_2 is not, it is the case that $f(Q, P_1) > f(Q, P_2)$.

Using the results of [TCC91], it can be shown that for any sound similarity function f, the problem of computing $f(Q, P)$, for a given Q and a given P, is NP-hard.

The reason for the high complexity given by the above lemma, is that in the worst case, one needs to examine all possible matchings between the objects in the query Q and the objects in the picture P; there are exponential number of such mappings as each object in the query can be mapped into multiple objects in the picture.

Maximal Matchings and Restricted Similarity Functions

In order to avoid the exponential complexity, we introduce another class of similarity functions, called *restricted similarity* functions. A restricted similarity function only considers a single *maximal* matching for computing the similarity value. A *maximal* matching is a matching that maps each object in the query to the closest object in the picture. The closest object in the picture to an object A in the query is defined to be the object whose "degree of closeness" with respect to A is maximum. Appendix B presents the definition of degree of closeness. Due to the characteristics of the inverse document frequency method of [Salt89] and the linear attributes that are employed in calculating the degree of closeness, it is highly unlikely that there are two or more closest objects in a picture corresponding to an object in the query. Thus, we assume that for any picture the maximal matching of the picture with respect to the given query is unique.

Formally, we define g to be a restricted similarity function if ,for any query Q and picture P, $g(Q, P) = g_1(Q, P) + g_2(Q, P) + g_3(Q, P)$, where g_1 is a number denoting how closely the objects in the query match with the objects in the picture, g_2 and g_3 respectively denote how closely the non-spatial and spatial relationships are satisfied ; all these numbers are computed with respect to the maximal matching. A detailed description on the computation of the numbers, given by g_1 and g_2, can be found in [ATY95]. We call g_3 to be the *spatial similarity* function and its value as the spatial similarity value.

5.1 Spatial Similarity Functions

In this subsection we discuss some of the properties that need to be satisfied by spatial similarity functions. It is to be noted that spatial similarity functions are computed with respect to the maximal matching.

For any set C of spatial relationships in a picture or a query, let $ded(C)$ denote the set of relationships deducible from C. Similarly, let $red(C)$ denote the reduction of C. For a query Q and picture P, we let $sat(Q, P)$ denote the set of spatial relationships in $ded(Q)$ that are satisfied by P with respect to the maximal matching. We say that a spatial similarity function h satisfies the *monotonicity* property if for any query Q and any two pictures P_1 and P_2 the following condition holds— if $sat(Q, P_1) \subseteq sat(Q, P_2)$ then $h(Q, P_1) \leq h(Q, P_2)$. The following similarity function satisfies the monotonicity property. It assigns a weight to each spatial relationship specified in the query or is deducible from the query, and computes the similarity value of a picture to be the sum of weights of the spatial relationships satisfied by it.

Now, consider the query Q specified in the following example (called example 1). The query Q specifies that object A is to the left of B, B is to the left of C , A is to the left of C, and D is above E. Suppose that there are two pictures, say P_1 and P_2. In P_1, the first three *left_of* relationships are satisfied, but the above relationship is not satisfied. In P_2, the first and the third *left_of* relationships, and the *above* relationship are satisfied but not the second *left_of* relationship. Both pictures satisfy 3 out of the 4 user specified relationships. If we use the above similarity function and assign equal weights to all the spatial relationships, then both the pictures in this example will have equal similarity values. However, it can be argued that the P_2 should have higher similarity value. This anomaly occurs because, when computing the similarity value of each picture, we did not distinguish those relationships that are fundamental, i.e. those in the reduction of the set of relationships satisfied by that picture, from those which are outside the reduction. For picture P_1, the first and the second *left_of* relationships are the only relationships in the reduction of the set of relationships satisfied by it. Where as for P_2, the reduction of the set of relationships satisfied by it consists of three relationships which are the first and the third *left_of* relationships and the *above* relationship. Suppose, for each picture, if we only consider those relationships in the reduction for computing similarity values, then P_2 will have higher similarity value than P_1.

We now construct a class of similarity functions, called *discriminating* similarity functions, that avoid the above anomaly and also satisfy the monotonicity property. The class of discriminating similarity functions work as follows. They first assign weights to the relationships in $ded(Q)$; recall that Q is the user query. For any picture P, they compute its similarity value to be the sum of the weights of all relationships in the set $red(sat(Q, P))$. Notice that these similarity functions ignore all the relationships satisfied by P that are

outside the reduction, because such relationships are directly implied by those in the reduction. It is easy to see that in example 1, if we give equal positive weights to all the relationships and use the above method, then the second picture will have a higher similarity value than the first picture.

To ensure monotonicity, when using discriminating similarity functions, we need to choose the weights of the relationships carefully. Consider the user query Q described below. In this query, all the "A" objects (i.e. $A_1, ..., A_n$) are to the left of B, and all the "C" objects (i.e. $C_1, ..., C_m$) are to the right of B. Now, consider two pictures P_1 and P_2 as given below. P_1 is identical to the query. P_2 has all the A and the C objects, but not B; all the A objects are to the left of all the C objects in P_2. It should be easy to see that $red(sat(Q, P_1))$ contains exactly $m + n$ relationships which are of the form A_i *left_of* B or B *left_of* C_j, while $red(sat(Q, P_2))$ contains mn relationships of the form A_i *left_of* C_j. Clearly, assignment of equal weights to all the relationships does not ensure monotonicity.

We now define a discriminating similarity function that satisfies the monotonicity property. Let Q be the user query. We define a directed graph $H = (V_H, E_H)$. The set of vertices V_H is exactly the set of relationships in $ded(Q)$ (here, for any pair of *overlaps* relationships of the form A *overlaps* B and B *overlaps* A, we have a single vertex in the graph). There exists an edge from the relationship r_i to r_j if there is a one step deduction of r_j that employs r_i. This graph denotes the generalization of the "uses" relationship defined in section 4. It can be shown that the graph H is acyclic. All the source nodes in the graph (i.e. nodes with no incoming edges) denote elements in $red(Q)$. Each vertex is assigned a level number as follows. The level number of any vertex r is the length of the longest path from any source node to v. The level number of a source node is zero, and the level number of any other node r is 1+ max{level number of $s : (s, r)$ is an edge in H}. The level numbers can be computed by a topological sort of H. We can assign arbitrary weights to all the source vertices, i.e. all the relationships in $red(Q)$. For all other vertices we assign weights inductively based on their level numbers. All the vertices having the same level number are assigned equal weights. Assume that there are k_i vertices at level i. For each level i node, assign a weight which is less than or equal to (the minimum weight of any vertex at level $(i-1))/(1+k_i)$.

6 Similarity Computations using Indices

Now, we describe two methods for retrieving the u most similar pictures using indices where u is given by the user. The key advantage of these indexed based methods is that they completely avoid examining those pictures that do not have any spatial relationship in common with the query. In these methods, we first compute the maximal matchings of the pictures with respect to the query. These matchings are captured as triples of the form (A, pid, oid) where A is an object in the query, oid is the id of the closest object to A in the picture pid.

After this we use indices on the spatial relationships. An index on a spatial relationship of the form U *op* V, where U and V are object types, contains the list of triples of the form $(pid, oid1, oid2)$ such that the objects with ids $oid1$ and $oid2$ in the picture given by pid are of the types U and V ,respectively, and they satisfy the spatial relationship op. The second method requires an additional flag to be present with each triple indicating whether the corresponding relationship is fundamental in the picture or not.

For a spatial relationship A *op* B in $ded(Q)$, we use the appropriate index to retrieve a list of triples of the form $(pid, aoid, boid)$ such that $aoid$ and $boid$ are the ids of the objects in pid to which A and B are mapped according to the maximal matching, and these objects satisfy the spatial relationship op in the picture pid. The $pids$ used in the two methods to be described are those appearing in these lists.

In the remainder of this section, we assume that the given user query Q specifies n objects and m spatial relationships.

6.1 First Method

The first method computes the similarity values for all pictures given by the relevant indices and picks those having the highest u similarity values.

For a given query Q, we first compute all the relationships in $ded(Q)$. (Recall that $ded(Q)$ is the set of all relationships deducible from Q). Let $r_1, r_2, ..., r_k$ be all these relationships. Next, we assign weights to each of these relationships as given in the previous section. Let w_i be the weight assigned to the relationship r_i.

For each r_i $(i = 1, ..., k)$, using the index on the spatial relationship, we retrieve a list L_i of all pids that satisfy r_i. We assume that each L_i is sorted in increasing values of the pids. We simply iterate the following steps for computing the similarity values for each picture present on one of the lists.

1. Let x be the smallest pid which is on top of all the lists. Now, it should be obvious that, the maximal matching in the picture x satisfies the relationship r_i iff x is on top of the list L_i. Using this, we compute $sat(Q, x)$ and delete x from all the lists in which it appears. For example, if x appears on

top of lists L_1, L_3 and L_5, then x satisfies exactly r_1, r_3 and r_5.

2. Using the reduction algorithms given in the previous sections, we compute $red(sat(Q, x))$, and compute the similarity value of x to be the sum of the weights of the relationships in $red(sat(Q, x))$.

Let p be the total number of all distinct pids appearing in any of the lists, and q be the sum of the lengths of all the lists. It should be easy to see that $q \geq p$. We can use a heap structure which contains the pids that appear on top of some list. The complexity of selecting the minimum and updating the heap is $O(\log k)$. The overall complexity of the above algorithm is $O(q \log k + pnk)$.

Let p be the total number of all distinct pids appearing in any of the lists, and q be the sum of the lengths of all the lists. It should be easy to see that $q \geq p$. We can use a heap structure which contains the pids that appear on top of some list. The complexity of selecting the minimum and updating the heap is $O(\log k)$. Hence the complexity of step 1, over all iterations, is $O(q \log k)$. To analyze the complexity we observe the following. To compute the reduction of a set containing at most k spatial relationships over n objects takes time $O(nk)$. Thus the complexity of step 2, over all iterations, can be shown to be $O(pnk)$. Hence, the overall complexity of the above algorithm is $O(q \log k + pnk)$.

The above algorithm can be made more efficient, by first observing that if a set S of multiple pictures satisfy the same set of relationships then we can simply calculate the similarity value for the first picture in S and store it in a table, and retrieve this value for the subsequent pictures in S by a table look up. By organizing this table as a binary tree of depth k (each level of the tree corresponds to a spatial relationship), searching and insertion can be done in $O(k)$ time. The overall complexity of this algorithm is $O(kp + lnk)$, where l is the number of different values of $sat(Q, P)$. Clearly, l is bounded by 2^k, which is the total number of subsets of $ded(Q)$, and $k = O(m^2)$. This method is better than the previous method, and also the method presented in the next subsection, when m and hence k is small.

6.2 Second Method

There are two phases in the second algorithm. Assume that the user is interested in retrieving the u most similar pictures, for some user-specified integer u. In the first phase, the algorithm estimates the spatial similarity value of each picture P having some spatial relationship in common with Q. In the second phase, the algorithm selects cu pictures with the highest estimated similarities for some multiplicative constant $c > 1$. The meta-data of these cu pictures will be further examined by the algorithm and the u most similar pictures among these cu pictures will be retrieved and presented to the user. As shown in the experimental results in Section 6.3, for $c = 3$, all of the u most similar pictures are retrieved for $u \leq 30$. The advantage of this algorithm is its low time complexity. Specifically, the time complexity is $O(q+cunk)$ where q and k are as given in subsection 6.1. Since cu is usually very small compared to q, (which is the total number of entries in all the inverted lists) , the first component dominates. Since all entries in the different inverted lists have to be examined by any algorithm, the proposed algorithm can be considered to be optimal within some multiplicative constant.

We now provide details of the two phases of the algorithm. For each picture P, $ded(P)$ is pre-computed and the spatial relationships of all the pictures are indexed. For each relationship $r \in ded(P)$, if $r \in red(P)$ then it is identified as fundamental (abbreviated as "f"), otherwise r is identified as a non-fundamental relationship (abbreviated as "nf"). This information is stored in the indices. Thus, if a relationship r is fundamental in picture $P1$ and is non-fundamental in picture $P2$, the index for r will contain the entries $(P1, f)$ and $(P2, nf)$. The following Lemma identifies two cases when a spatial relationship in $sat(Q, P)$ is guaranteed to be in $red(Sat(Q, P))$.

LEMMA 6.1: For any $r \in Sat(Q, P)$, if $r \in red(Q) \cup red(P)$, then $r \in red(Sat(Q, P))$.

During the first phase, we examine and process each entry on the inverted list associated with each relationship r in $ded(Q)$, as follows. If the entry corresponds to picture P, and either r is fundamental in Q or in P, then we simply add the $weight(r)$ to the estimated similarity value of P (in this case r is in $red(Sat(Q, P))$ by lemma 6.1); If the above condition is not satisfied, i.e. r is not in $red(Q) \cup red(P)$, then the probability p that r is in $red(Sat(Q, P))$ is estimated, and the similarity value of P is increased by $p \cdot weight(r)$. A particular heuristic for estimation of p is given in 6.3.

In the second phase of the algorithm, the meta-data associated with the cu pictures with the largest estimated similarities are retrieved. For each such picture P, the exact similarity value of P is computed by obtaining $red(Sat(Q, P))$.

6.3 Experimental Results

We now present some experimental results obtained using the second method. In the experimentation we have used a weight of one for each spatial relationship.

The first phase of the implemented algorithm has two steps. In the first step the algorithm goes through

all the inverted lists, and for each picture P appearing in at least one of these lists, it computes the following numbers—
N_p = the number of relationships in $Sat(Q,P)$ which are fundamental relationships in P;
N_q = the number of relationships in $Sat(Q,P)$ that are fundamental in Q and non-fundamental in P;
N_* = the number of relationships in $Sat(Q,P)$ that are non-fundamental in both P and Q;
M = the number of relationships in $ded(P)$;
M_f = the number of fundamental relationships in P.

In the second step of the first phase, the size of $red(Sat(Q,P))$, denoted by n_0, is estimated. Since the weight of each relationship is taken to be one, the similarity value for each picture P which is in at least one of the inverted lists is simply taken as n_0. According to lemma 6.1, each relationship in $Sat(Q,P)$ which is fundamental in either P or Q, is in $red(Sat(Q,P))$. Due to the above reason, we initialize n_0 to be N_p+N_q. After this, n_0 is updated as follows. Here the algorithm makes N_* iterations. Each of these iterations corresponds to a relationship r in $Sat(Q,P)$ which is non-fundamental in both P and Q. In each such iteration, a probability p that the relationship r in $Sat(Q,P)$ is in $red(Sat(Q,P))$ is estimated. The following rule is used for computing p and updating n_0.
If $N_p = M_f$ then $p := 0$;
else $\{p := max\{0,\ 1-N_p/M_f-\alpha n_0/M\};\ n_0 := n_0+p;\}$

In the above computation, α is taken to be 0.1. The justification for using the above formula for estimation of p is the following. The first half of the rule says that, if all fundamental relationships in P are in $Sat(Q,P)$ then any non-fundamental relationship in P that is in $Sat(Q,P)$ must be outside $red(Sat(Q,P))$. The second half of the rule ensures that p decreases with increasing values of N_p and n_0.

We have carried out experiments to determine the maximum number of pictures that need to be retrieved using the estimated spatial similarity values so as to capture u most similar pictures. The experimentation was carried out on a randomly generated meta-data for 10,000 pictures. Objects in a picture were randomly chosen from a set of 50 objects. The number of objects in a picture is chosen to be between 5 and 16. We only used the relationships *inside*, *overlaps* and *above* in each picture (*left_of* and *in_front_of* are orthogonal to *above* and can be handled in the same way. *outside* relationships are of less significance and are not considered in the experiments.). The relationships in a picture were randomly generated and the number of relationships was varied from 3 to 144. Ten queries were randomly generated with the number of objects in each query being between 4 and 15, and the number of relationships varying from 3 to 90.

u	Q_1 4,2,1	Q_2 4,3,3	Q_3 8,7,1	Q_4 7,6,12	Q_5 10,10,30
10	10	10	10	10	30
20	20	20	20	36	45
30	30	30	30	40	50
40	40	40	40	41	62
50	50	50	50	85	67

u	Q_6 10,10,31	Q_7 10,11,30	Q_8 10,12,22	Q_9 14,18,63	Q_{10} 15,23,67
10	29	10	16	10	10
20	29	32	23	22	47
30	58	32	33	31	51
40	59	166	52	140	68
50	62	167	60	186	74

Table 6.1 Performance of Method 2

The results of the experiment are given in table 6.1. Each row corresponds to a particular value of u given in the first column. Each column, from the second column onwards, denotes a query Q_i, with parameters a, b, c that denote the number of objects, the number of fundamental and non-fundamental relationships in the query. An entry under the column of Q_i, denotes the maximum number of pictures that have to be retrieved for that query using the estimated similarity values, so as to capture all the u most similar pictures according to the actual similarity values.

It can be seen that, for $u = 10, 20, 30$, method 2 only needs to retrieve top $3u$ pictures based on the estimated similarity values. For these $3u$ pictures the actual similarity values are computed by retrieving their meta-data. It should be noted that for most practical purposes, a user would like to see no more than 30 most similar pictures, and therefore method 2 will be effective.

7 Conclusions and Discussion

In this paper, we have presented two methods for retrieving pictures using similarity based approaches that use indices on spatial relationships of objects. We have implemented the second method and analyzed its performance using a medium sized randomly generated experimental database of pictures. The preliminary results of this experimentation are very encouraging.

We have presented efficient algorithms for the deduction and the reduction problems. These algorithms are of independent interest for the following applications. When meta-data need to be transmitted from one site to another, the cost of transmission can be minimized by simply transmitting the minimal reduction. In this case, the implied relationships can be recovered from the minimal reduction by using the deduction algorithm. The minimal reduction can also be

very useful for *exact match* retrieval, i.e. for retrieving pictures that satisfy all the spatial relationships in the query. In this case, it is enough to retrieve pictures that satisfy all the relationships in the minimal reduction of the query. This saves execution time.

We have developed/are developing four prototype picture retrieval systems. These are used for retrieving photographs [ATY95], for skin cancer detection and for buying clothing and parts respectively.

References

[Amd93] Amdor, F.G. et al., *Electronic How Things Work Articles: Two Early Prototypes*, IEEE TKDE, 5(4), Aug. 1993.

[ATY95] A. Aslandogan, C. Thier, C. T. Yu, et al "Implementation and Evaluation of SCORE(A System for COntent based RE-trieval of Pictures)", IEEE Data Engineering Conference, March 1995.

[Car93] Cardenas, A.F. et al.: The Knowledge-based Object-Oriented PICQUERY+ Language, IEEE TKDE, 5(4), Aug. 1993.

[CCT94] S.K. Chang, G. Costaliola and M. Tucci, *Representing and Retrieving Symbolic Pictures by Spatial Relations*, to appear in J. of Visual Language and Computing.

[ChH92] Chang, S.K., and Hsu,A., *Image Information Systems:Where Do We Go from Here?*, IEEE TKDE Vol. 4,No. 5.

[CK81] Chang S K, Kunii T L: Pictorial Data-Base Systems, IEEE Computer, Nov 1981.

[CSY84] Chang, S. K., Shi, Q.Y. and Yan, C.W. *Icomic Indexing by 2-D Strings.* IEEE Transactions on Pattern Analysis and Machine Intelligence, July 1984.

[Ch92] Chu, W. et al, *A Temporal Evolutionary Object-Oriented Model and its Query Languages for Medical Image Management*, VLDB 92.

[Eg89] Egenhofer M. J., *A Formal Definition of Binary Topological Relationships*, in W. Litwin and H. J. Schek, editors, FODO 89, France.

[GZCS94] Gong Y. et al, *An Image Database System with Content Capturing and Fast Image Indexing Abilities* IEEE Multimedia Conference, 1994.

[GrM92] Grosky W. and Mehrotra, R., *Image database Management*, Advanced in Computer, Vol. 34, Academic Press, New York.

[GR94] V. N. Gudivada and V. V. Raghavan, *Design and evaluation of algorithms for image retrieval by spatial similarity*, ACM T. on Inf. Systems, 1995.

[GWJ91] Gupta, A., Weymouth, T., and Jain, R. *Semantic Queries with Pictures: The VIMSYS Model* VLDB 91.

[HOP91] S.A. Hawamdeh, B. Ooi, R. Price, T. Tang, Y. Deng and L. Hui, *Nearest neighbor searching in a Picture Archive System*, Int. Conf. on Multimedia Inf. Systems, 1991.

[LeeW93] Lee, Eric and Thom Whalen, *Computer Image Retrieval by Features: Suspect Identification*, ACM Conf. on Human Factors in Computing Systems, 1993.

[LSY89] S. Y. Lee, M. K. Shan, and W. P. Yang, *Similarity retrieval of* ICONIC *image databases*, *Pattern Recognition*, 1989.

[Ni93] Niblack W. et al, *The QBIC Project: Querying Images by Content Using Color, Texture and Shape*, IBM Report, 1993.

[RP92] Rabitti, F and P Savino, *An Information Retrieval Approach for Image Database*, VLDB 92.

[RM89] Reiter R., Mackworth A. K., *A Logical Framework for Depiction and Image Interpretation*, Artificial Intelligence 41, 1989.

[Salt89] Salton G. : "Automatic Text Processing", Addison Wesley, Mass., 1989.

[SYH94] A. Prasad Sistla, Clement Yu, R. Haddad, *Reasoning About Spatial Relationships in Picture Retrieval Systems*, VLDB 94.

[TaY84] Tamura, H. and Yokoya, N., *Image Database System:A Survey* Pattern Recognition, Vol. 17, No. 1.

[TCC91] M. Tucci, G. Costagliola and S.K. Chang, *A Remark on NP-completeness of Picture Matching*, IPL 1991.

8 Appendix A: Rules for Deducing Spatial Relationships

I. (Transitivity of 'left_of', 'above', 'behind', and 'ins

For each relation x in {*left_of*, *above*, *behind*, *inside*}, we have the following rule:
$A\ x\ C$ :: $A\ x\ B$, $B\ x\ C$

II. This rule captures the interaction between the relationships involving *left_of*, *above*, *behind*, and the relationship involving *overlaps*. For each x in {*left_of*, *above*, *behind*}, we have the following rule.
$A\ x\ D$ :: $A\ x\ B$, B *overlaps* C, $C\ x\ D$

III. This rule captures the interaction between the relationships involving *left_of, above, behind, outside*, and the relationship involving *inside*. For each x in {*left_of*, *above*, *behind*, *outside*}, we have the following rules.
(a) $A\ x\ C$ :: A *inside* B, $B\ x\ C$
(b) $A\ x\ C$ :: $A\ x\ B$, C *inside* B

IV. (Symmetry of *overlaps* and *outside*)
This rule captures the symmetry of *overlaps* and *outside*. Let x denote either of *overlaps* and *outside*. We have the following rule for each such x.
$A\ x\ B$ :: $B\ x\ A$

V. This rule allows one to deduce that two objects are outside each other if one of them is to the left of, or above, or behind the other object. Let x denote any of the relationship symbols in {*left_of*, *above*, *behind*}. We have the following rule for each such x.
A *outside* B :: $A\ x\ B$

VI. This rule allows one to deduce that if an object is inside another object, then the two objects overlap.
A *overlaps* B :: A *inside* B

VII. This rule allows one to deduce that A overlaps with B if B overlaps with an object inside A.
A *overlaps* B :: C *inside* A, C *overlaps* B

VIII. This rule says that every object is inside itself. Note that this rule has no body.
A *inside* A ::

9 APPENDIX B

In this section we briefly describe how the degree of closeness , mentioned in section 5, of an object B in the picture with respect to A is defined and calculated. A detailed description for computing the degree of closeness of two objects is given in [ATY95]. Let object A in the query have values $u_1, ..., u_k$ for the attributes $X_1, ..., X_k$ respectively, and let object B in the picture have values $v_1, ..., v_k$ for the same attributes respectively. The degree of closeness of A and B is defined as $(\Sigma_{1 \le i \le n} w_i)$, where w_i is a number denoting how closely the attribute values u_i and v_i match and is computed as follows.

If $u_i = v_i$ then w_i is given by the inverse document frequency method of [Salt89]; roughly speaking, in this method, the value of w_i is higher if there are fewer pictures in the database that have an object whose value for the attribute X_i is v_i. This method has been verified to yield better retrieval accuracy than that of assigning equal weights to all values of the attribute X_i [ATY95].

In the other cases, w_i is determined as follows. If the values for the attribute X_i can be arranged in a linear order so that neighboring values can be semantically similar then we call X_i to be a *linear attribute*. For example, the attribute *age* takes values *very young, young, middle aged, old* and *very old*, and it is a linear attribute; here, for example, the neighboring values *very young* and *young* are semantically similar. If X_i is a linear attribute, and u_i, v_i are neighboring values then w_i is given by a fraction of the weight obtained by the inverse document frequency method. If X_i is a linear attribute and u_i, v_i are not neighboring values then w_i is $-\infty$; this implies that objects A and B cannot be matched. If X_i is not a linear attribute (ex. *profession* is one such attribute) and $u_i \neq v_i$ then $w_i = 0$.