Efficient Position-Independent Iconic Search Using An R-Theta Index

Charles B. Cranston and Hanan Samet *
Center for Automation Research, Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland, College Park, MD 20742
zben@cs.umd.edu,hjs@cs.umd.edu

ABSTRACT

An iconic image database is a collection of symbolic images where each image is a collection of labeled point features called icons. A method is presented to support fast position-independent similarity search in an iconic database for symbolic images where the similarity condition involves finding icon pairs that satisfy a specific spatial relationship. This is achieved by introducing an index data structure based on r-θ space, which corresponds to the Cartesian product of separation (i.e., inter-icon distance) and (some representation of) relative spatial orientation. In this space, each pairing of two icons is represented by a single point, and all pairs with the same separation and relative orientation (regardless of absolute position) map to the same point. Similarly, all icon pairs with the same separation but different relative orientations map to points on a line parallel to the θ axis, while all pairs with different separations but the same relative orientation map to points on a line parallel to the r axis. Using such an index, database search for icon pairs with a given spatial relationship or range is accomplished by examining the subarea of the index space into which desired pairs would map. This r-θ index space can be organized using well-known spatial database techniques, such as quadtrees or R-trees. Although the size of such an index grows only linearly with respect to the number of images in the collection, it grows quadratically with the average number of icons in an image. A scheme is described to reduce the size of the index by pruning away a subset of the pairs, at the cost of incurring additional work when searching the database. This pruning is governed by a parameter ϕ, whose variation provides a continuous range of trade-offs between index size and search time.

* The support of the National Science Foundation under Grants EIA-00-91474 and CCF-0515241, Microsoft Research, and the University of Maryland General Research Board is gratefully acknowledged.

ACM-GIS'06, November 10–11, 2006, Arlington, Virginia, USA.

Categories and Subject Descriptors: H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing—*Indexing methods*

General Terms: Algorithms

Keywords: iconic database, position-independent, spatial indexing

1. INTRODUCTION

Maps provide a means of maintaining a database of cartographic information. Although maps are usually thought of as capturing positional information, they are also used to capture spatial relation information. In particular, relevant objects or features are represented by symbols, and both the topological layout of these objects and their inter-object distances are significant. Examples of such maps include floor plans, blueprints, and satellite images. Examples of features include doorways and windows, component parts of an assemblage, and rivers, roads, and buildings.

A collection of digitized map images is an example of an image database. Searching such a database includes the determination of which images within the database contain a desired arrangement of symbols, such as a beach north of a city or a youth hostel near a train station. For such a search, the absolute position of the symbols within the database image is unimportant, only the relative spatial relationship of the symbols is significant. This is called a *position-independent search*. Search for symbols with a particular spatial relationship has applications in mobile computing, mobile data management, moving-object and moving-framework databases, and location-based services.

One method of specifying such a search is by constructing a *query image* containing the desired topological arrangement of symbols (*icons*). If database images can contain multiple instances of a particular kind of icon (such as multiple hotels in the same geographic area), an icon in a query image actually designates (a member of) an *icon class*, to be satisfied by a corresponding specific icon *instance* in database images matched. In addition to a topological arrangement of icon class designators, a query must also specify which spatial relationships between the classes of the query image are to be considered relevant in the search. This describes a spatial version of *retrieval by content*, which has been investigated in both the spatial and non-spatial domains. The MARCO (MAp Retrieval by Content) system of Samet and Soffer [21, 25] is an example of this search method.

While a minimal database search capability can be constructed using little more than a basic similarity measure (as a brute-force search would simply compare the query image against each database image in turn), efficient database access usually depends upon an *index*, an auxiliary data structure that is maintained by the database system to allow the efficient determination of (and thus access to) only the portion of the database relevant to the current task. Indexes often embody *abstraction* (with only a subset of the data being replicated into the index), and *structure* (supporting access to the index entries in more sophisticated ways than simply in sequential order of the database proper). The simplest form of such structure is an *ordering*. For example, in a book index, only certain key words are abstracted into the index, which is then alphabetically ordered. However, more complicated structuring methods are also possible.

As it is inefficient to regenerate the index for each new task, the information in the index must be in some sense *invariant*, that is, useful (perhaps to a greater or lesser extent) for all supported tasks. While many methods of generating an invariant may be possible, it is also necessary that the method chosen be a useful one. For example, in a database of two-dimensional line segments, one choice for an invariant would be a single point in four-dimensional space, with the two-dimensional coordinates of both end points supplying the four required coordinates. This is known as the *corner transformation* [22], which is a two-dimensional variant of an interval representation (e.g., [5]) and used by a number of researchers (e.g., [6, 11, 12, 15, 19, 26]). However, this invariant is not always useful in satisfying important queries about line segment intersections or proximity. In this particular case, an invariant involving partitioning underlying two-dimensional space and then indexing the partitions intersected by each line segment is preferable for answering such queries.

Although much of the image database research has examined matching a single query icon to a database icon, some prior work [1, 2, 3, 4, 7, 8, 9, 13, 14, 16, 17, 18, 23, 24] does address queries containing several icons. One approach [3, 14, 16] involves determining the projections of image objects onto the coordinate axes and then encoding these projections as strings. The resulting encoded strings serve in turn as invariants for the index.

Another approach involves modeling the arrangement of image objects as an Attributed Relational Graph (ARG). In [2] a metric-space index is described, using a distance metric function based on the *optimal error-correcting (sub)graph isomorphism problem*. While this is an example of an index structure that is more complex than simple ordering, the algorithm given for computation of the exact distance between two ARGs requires "polynomial time of the fifth order". Furthermore, as the ARG used to place each database image into the index is evaluated using all objects present in that image, the efficiency of the method given degrades when the query ARG represents fewer objects than are present in database images. However, their approach does support fuzzy identification of database objects.

Gudivada and Raghavan [8, 9] have investigated search based on the relative spatial relationships of icon pairs. The algorithm first determines the n icons common to the query and database images, and then compares the slopes of the $n(n-1)/2$ lines between icon pairs in each database image to the slopes of the lines between corresponding icon pairs in the query image, thus abstracting away the absolute positions of the icons and considering only the relative spatial relationships, as captured by these line slopes. Furthermore, by clustering the set of line slopes, their algorithm can perform a rotation-independent search. Their approach does generate a measure of the similarity between a particular query image and a particular database image. However, because the common icon set depends on the particular query image, their algorithm does not generate an invariant that could be used as the basis of a database index.

In contrast, the search algorithms to be described here use an index created by examining every image in the database, and mapping each pairing of icons into a single point in an abstract space consisting of the Cartesian product of separation (inter-icon distance) and relative orientation. We call this an r-θ space. The absolute positions of the icons are abstracted away, the icon separation of the pair determines the r coordinate, and the relative orientation determines the θ coordinate. All pairs with the same separation and relative orientation (regardless of absolute position) map to the same point in this space. All pairs with the same separation but different relative orientations map to points on a line parallel to the θ axis, while all pairs with different separations but the same relative orientation map to points on a line parallel to the r axis. This r-θ index can be organized using well-known spatial database techniques, such as quadtrees [20] or R-trees [10]. This approach provides a position-independent search, but does not provide a rotation-independent search,

Because an image of n icons contains $n(n-1)/2$ pairs, the size of such an index grows quadratically with the number of icons in an image (although only linearly with the number of images in the database). For databases with only a sparse population of icons this may not be a severe problem. Another approach is to *prune* some of the pairs from the index, at the cost of requiring some additional work while searching the index.

The remainder of this paper is organized as follows: Section 2 shows the structure and construction of an unpruned version of an index. Section 3 discusses various kinds of queries and the spatial constraints they imply, and how these constraints determine the part of the area spanned by the index to access. Section 4 provides a method of pruning the index to reduce its size. Section 5 outlines modifications to the search algorithm required to support such pruning. Section 6 gives the results of Monte-Carlo simulation experiments designed to determine the efficacy of index pruning. Section 7 describes a variant where separate indexes are kept for each pair of icon types. Section 8 discusses considerations for updating an index when the database is changed. Section 9 discusses how such an index can be used for non-position-independent search. Section 10 briefly shows how such an index can be used for purely existential searches, which do not invoke any spatial constraints. Section 11 contains concluding remarks.

2. NON-PRUNED INDEX CONSTRUCTION

The relative spatial orientation of two icons can be characterized by the slope of the line connecting them, or alternatively can be characterized as an angle. Because the slope of a vertical line is mathematically infinite, it is more tractable to characterize this orientation as an angle. Therefore, the θ axis of the r-θ space is most conveniently calibrated in some angular measure.

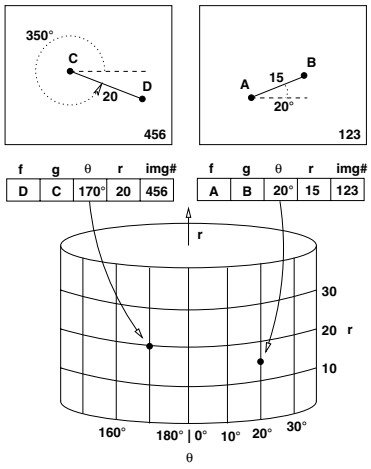


Figure 1: Mapping of icon pairs into r-θ space

Pairs of icons possess a reflection symmetry: if, with respect to icon A, there exists an icon B at a particular distance and relative orientation, there also exists, with respect to icon B, an icon A at the same distance and the "opposite" orientation. This symmetry can be used to halve the number of entries in the index, if it is possible to determine, when searching for an icon pair of classes A and B, whether to search for an A with a particular orientation towards a B or instead for a B with the "opposite" orientation towards an A. One way to address this (except for the special case of searching for an A with respect to another A) would be to impose a total order on the icon classes. However, in the work described here the spatial orientation of two icons is limited to the range 0° to 180° by exchanging the icons if their orientation lies outside this range.

The choice between these two methods of halving the size of the index is another trade-off between index size and search time. Roughly speaking, throwing the same number of points into a smaller space results in an increased density of points, so in the pruning algorithm of Section 4 this higher density increases the probability that evidence to enable pruning will be found, thereby decreasing the size of the resulting index. However, when searching with such an index, twice the number of entries must be examined.

The r-θ space that we use is not a traditional Euclidean space because it is a two-dimensional space that is half-open in one dimension (r) but circularly closed in the other (θ) dimension. More precisely, the r dimension starts at zero and theoretically runs to infinity, although in any particular database there will be a maximally separated icon pair, which will then determine the maximum r coordinate value for that database. The θ dimension is topologically closed, that is, 0° and 180° are logically identical. Thus the space is a half-cylinder running from zero to infinity.

Figure 1 illustrates the mapping of two icon pairs from database images 123 and 456 into r-θ space. The relative orientation of icons A to B from database image 123 is 20°, which is less than 180°. Since icons A and B are 15 units apart, the pair maps into the point $r = 15$, $\theta = 20$. In database image 456, icons C and D are 20 units apart, but because the relative orientation of C toward D of 350° is greater than 180° the two icons are exchanged and the pair maps into the point $r = 20$, $\theta = 170$.

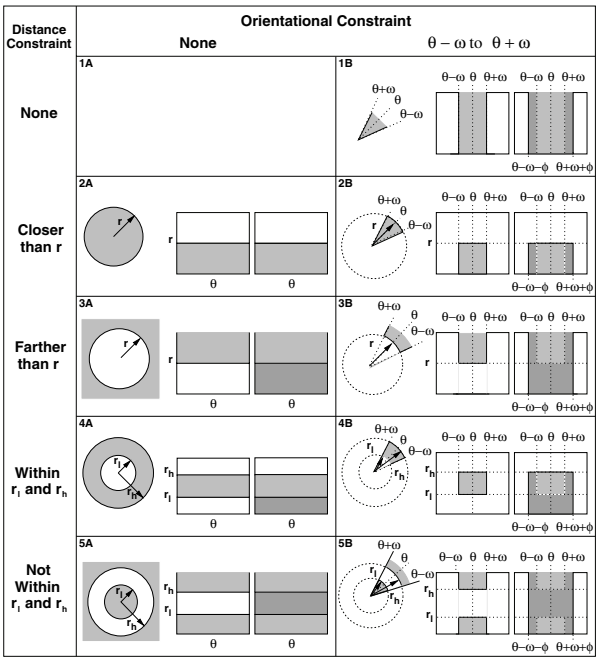
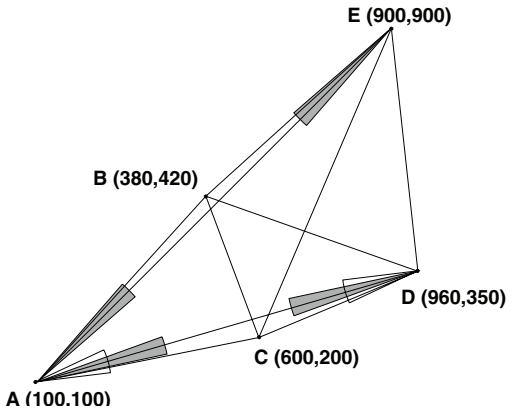


Figure 2: Areas of r-θ index space to be searched

Construction of the non-pruned index is not complicated. For each symbolic image in turn, each pairing of icons is considered, and an index entry for that pairing is added. This entry must explicitly contain identifiers for the two icons (perhaps including icon class), and an identifier for the particular symbolic image, which is used to indicate the images found by a successful query. In addition, given the index entry, it must be possible to recover both the separation and relative orientation of the pair, although this data may be implicit in the spatial data structure that is used. One possibility is to explicitly keep separate Δ_x and Δ_y (relative distance in x and y) for the pair, as computation and comparison of separation can be done in the r^2 domain and relative orientation can be recovered by using a four-quadrant arctangent function such as Fortran's ATAN2.

3. NONPRUNED INDEX SEARCH

When searching the index, the set of applicable spatial constraints determines an area of the r-θ index that must be examined. The 9 nontrivial boxes in Figure 2 represent different combinations of separational and orientational constraints. The boxes labeled A represent distance constraint, while the boxes labeled B represent a combination of both distance and orientational constraints. For the orientational constraints described in this figure, ω represents the range of orientations desired, that is, icon pairs with orientations between $\theta - \omega$ and $\theta + \omega$ are sought. In each box, the left image shows a representation in normal 2-D space of the target of such a query, and the middle image shows the area of r-θ space examined. (The right image shows the search area for a pruned index, described below). For example, in case 5A, the search is for icon pairs either closer together than r_l or farther apart than r_h. In this case the area of r-θ space examined is the band below r_l and the band above r_h. In case 4B, where separation and orientation are both con-



Pair	r	Raw θ	Effective θ
AB	425	48.8°	
AC	509	11.3°	
AD	895	16.2°	
AE	1131	45.0°	
BC	311	315.0°	135.0°
BD	584	353.1°	173.1°
BE	707	42.7°	
CD	390	22.6°	
CE	761	66.8°	
DE	553	96.2°	

Figure 3: AB and BE constitute evidence for AB

strained, the examined area is a rectangle (or window), and various well-known methods can be used for such a window query.

4. PRUNED INDEX CONSTRUCTION

The intuition for index pruning is the observation that some entries, if included in the index, can act as proxies or *evidence* for others, allowing the latter to be omitted from the index. However, upon search within such a pruned index, the area of r-θ space that must be examined is generally larger than the area required for an unpruned index. Post-processing of the retrieved index data is also required.

Figure 3 shows an example database image containing five icons, and the ten corresponding icon pairs along with their r and θ values. The θ values of pairs BC and BD are above 180°, so both pairs will be reversed, yielding effective θ values of 135.0° and 173.1°.

The two gray cones at points A and E are both 8° wide, corresponding to a ϕ value of 4°. In an index pruned to this ϕ value, pairs AB and BE constitute evidence for pair AE, because their orientations (θ values) of 48.8° and 42.7° differ from the 45° orientation of pair AE by less than the pruning parameter 4°. Neither pairs AD and DE nor pairs AC and CE constitute evidence for AE. Pairs AC and CD do not constitute evidence for AD. However, for an index pruned to a ϕ value of 10° (open cone), AC and CD do constitute evidence for AD.

In the general case, determining the minimal subset of pairs providing complete evidence for all pairs (and thus comprising the smallest possible index for that ϕ value) may be of high-order time and space complexity, and therefore best addressed by the techniques of dynamic programming.

Alternatively, there are a wide range of heuristic evidence

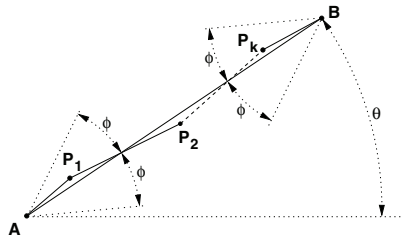


Figure 4: Evidence for pair AB. Note all slopes lie between $\theta - \phi$ and $\theta + \phi$.

strategies that could be employed. Because the search algorithm itself is used during index pruning, soundness of this class of indexing schemes is guaranteed if the evidence examined at search time is a superset of the evidence considered at the time the index is pruned.

The strategy used to build and search the indexes described in this paper is based on two design choices. First, evidence for an entry with a larger r value (representing an icon pair that is farther apart) will be sought only among entries with smaller r values (pairs that are closer together). Second, evidence for an entry with a given θ value will be sought only among entries whose θ values (relative orientation) differ from it by at most a search width parameter ϕ. For an index pruned to larger values of ϕ, evidence is sought among a larger number of entries, (in general) more entries can be pruned, and eventually a smaller index is generated.

The intuition for these design choices is as follows. For any given icon pair that a particular query might accept, that pair may or may not have been pruned from the index. The area of the index space where it would (if not pruned) be present comprises the minimal area that must be accessed. One way to reduce the total area of the index space accessed is to limit the area examined for evidence to as small an extension of this minimal space as possible. Under these design choices, the area accessed is extended down to the $r = 0$ axis and (as we shall see) widened by the pruning parameter ϕ. Other choices are possible. For example, if evidence for pairs with a separation of r is sought only within pairs with separations from r to $r/3$ then the search area need only be extended down to $r/3$. However, in this case less pruning may be possible.

An index entry for a particular icon pair may be pruned if and only if implicit evidence for that pair can be found by the search algorithm. In Figure 4 the evidence for an icon pair AB is shown. This evidence consists of a set of explicit pairs in the index comprising a sequence $A-P_1, P_1-P_2, \ldots, P_k-B$ that connects A and B; and that the relative orientation (angle) of each of these pairs is within ϕ of that of the original pair AB (this relative orientation constraint limits the extent in the θ dimension of index space that must be accessed). The Union-Find algorithm, operating on the pairs found in the index, is used to make the association between A and B in slightly more than linear time.

The pruned index is constructed by the following greedy algorithm. Independently for each image in the database, all icon pairings are generated and sorted on ascending r (closer to farther). For each pairing AB considered in this order, the entries already included in the index are filtered according to the ϕ constraint and the Union-Find algorithm is run on the result. If evidence for AB is not found, an

r	θ	Unpruned Index	Pruned $\phi = 4°$	Pruned $\phi = 10°$
311	135.0°	CB	CB	CB
390	22.6°	CD	CD	CD
425	48.8°	AB	AB	AB
509	11.3°	AC	AC	AC
553	96.2°	DE	DE	DE
584	173.1°	DB	DB	DB
707	42.7°	BE	BE	BE
761	66.8°	CE	CE	CE
895	16.2°	AD	AD	
1131	45.0°	AE		

Figure 5: Unpruned and Pruned Indexes

r	θ	Indexed	Selected
311	135.0°	CB	
390	22.6°	CD	CD
425	48.8°	AB	AB
509	11.3°	AC	
553	96.2°	DE	
584	173.1°	DB	
707	42.7°	BE	BE
761	66.8°	CE	
895	16.2°	AD	

Figure 6: Search of a $4°$ index for pairs at $34° \pm 12°$

entry for AB is added to the growing index. In Figure 5, the entries of pruned indices for the icons of Figure 3 are shown for pruning factors of $\phi = 4°$ and $\phi = 10°$. In either case, when pair AE is considered by the algorithm, evidence for AE (consisting of pairs AB and BE) is found, consequently AE is not added to the index. Similarly, when the $\phi = 10°$ index is built, AD is not added to the index, as AC and CD serve as evidence.

Generating the full set of pairings and sorting them in ascending order of separation requires $O(n^2)$ storage space and $O(n^2 \log n)$ execution time. As each of the $O(n^2)$ pairs is considered, the subset of pairs already in the index and within the orientational constraint must be examined. As there will be between $O(n)$ and $O(n^2)$ of them, construction of the index requires between $O(n^3)$ and $O(n^4)$ time,

5. PRUNED INDEX SEARCH

When searching the index, all pairs in an appropriately expanded area of the index are retrieved. In order to ensure that a pruned pairing can always be found, the pairs retrieved by the search algorithm must include all evidence that might have been considered by the original pruning decision. This requires accessing a somewhat larger area of r-θ space, relative to that examined by the same search on an unpruned index.

In the index of Figure 6 a search for pairs with orientations between 22° and 46° ($\theta = 34° \; \omega = 12°$) should find pair AE. Because the evidence for AE includes AB at 48.8°, the area of the index retrieved must be expanded in the θ dimension to the range 18° ($\theta - \omega - \phi$) to 50° ($\theta + \omega + \phi$). (Note that CB is not selected. Although its 135.0° orientation appears to be a variant of 45.0°, it is in reality directly opposite.)

In Figure 7 (I) the evidence considered by the original pruning decision for pair $A_k - B_k$ includes pairs whose relative orientations lie between $\theta_k - \phi$ and $\theta_k + \phi$. This is shown in the θ dimension of r-θ space (II). When searching for a pair with relative orientation within ω of θ (III) the evidence for pairs close to the edge of the search criterion may protrude as far as ϕ beyond that space, so the area accessed in r-θ space must be widened by the index pruning factor ϕ in order to include this evidence. This is shown in r-θ space in (IV). The additional area that must be accessed is shown in dark gray.

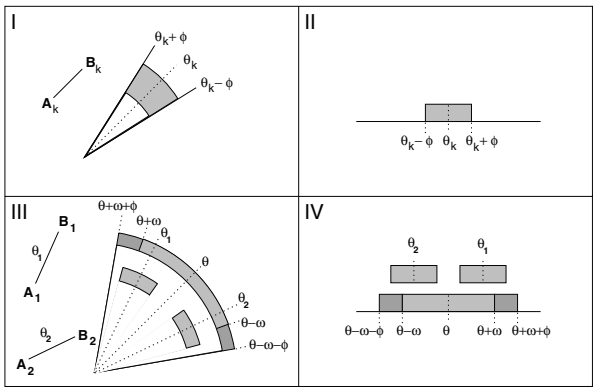


Figure 7: Search in θ dimension broadened by ϕ

In each of the five boxes 1B through 5B on the right half of Figure 2, the rightmost illustration shows widening of the accessed space in the θ dimension by ϕ, the pruning factor of the index being searched.

In addition, because closer pairs act as evidence for farther pairs, the area of r-θ space accessed must be extended down to the $r = 0$ axis. In cases 1B, 2A, 2B, 5A, and 5B in Figure 2 this area down to the $r = 0$ axis is already accessed, so the extension of search in the r dimension imposes no additional cost, while in cases 3A, 3B, 4A, and 4B some additional cost is incurred. In all boxes of this figure the additional area that must be accessed due to considerations of index pruning are shown in dark gray.

The evidence retrieved from the implicated area of r-θ space is processed by the Union-Find algorithm. Each icon pair retrieved is interpreted as signifying an equivalence relation between its two constituent icons, thus a set of equivalence classes of icons (or *clusters*) is the final result.

If, at index creation time, an icon pair A-B was pruned from the index, the evidence examined at that time must have contained the chain of pairs $A-P_1, P_1-P_2, \ldots, P_k-B$, all within the constraints on relative orientation. Because of the extension of the accessed space in both the r and θ axes, the retrieved entries are guaranteed to contain all this evidence, and the chain of pairs will cause the Union-Find algorithm to place icons A and B together in the same output cluster.

Each cluster generated by the Union-Find algorithm is examined separately for pairs satisfying the search constraints. After filtering the icons in a particular cluster by icon class, each pairing of the remaining icons is examined. In the example of Figure 6 clusters (ABE) and (CD) are generated, and the final results AE, BE, and CD are found.

The time complexity for this phase rises quadratically with cluster size, but is somewhat mitigated by two factors. First, the algorithm is actually quadratic on the cluster size;

		Pruning Parameter ϕ			
n	$\frac{n(n-1)}{2}$	2	4	6	8
10	45	40	38	35	35
20	190	160	142	126	105
30	435	328	277	234	194
40	780	552	428	347	286
50	1225	776	588	464	385
60	1770	1036	749	573	477
70	2415	1299	907	688	578
80	3160	1567	1068	809	666
90	4005	1844	1244	937	758

Figure 8: Sizes of Unpruned and Pruned Indexes

thus processing of a large number of small clusters will require less time than processing a small number of large clusters. Second, each cluster is filtered based on icon class, so the size of the data is greatly reduced before the quadratic phase of the algorithm.

Note that if a distance constraint $r < 400$ had been specified, pairs AB and BE would not have been retrieved, and only pair CD would be present in the result. If a distance constraint $r < 1000$ had been specified, pairs AB and BE would still have been retrieved, but AE would have been filtered out by the final pass, and again, only pair CD would be present in the result.

6. SIMULATION RESULTS

Monte Carlo simulation is a methodology that uses a random number generator to construct a set of test cases, along with a statistical analysis of the results of those tests. A Monte Carlo simulation was undertaken to investigate the effectiveness of pruning in reducing index size, and the degree to which clustering reduces the execution time of the search algorithm.

If the full circle is divided into n units (such as 360 degrees), then modulo-n arithmetic can be used for computation on this axis. Furthermore, if n is chosen to be a power of two, a bitwise AND can be used to implement the modulo division operation. In the simulations that we ran, the circle was divided into 256 (2^8) units.

The version of the Union Find algorithm implemented for this simulation was extended to preserve the offsets Δ_x and Δ_y between each icon and the *representative* chosen by the algorithm for each equivalence class (cluster) it produces. The offsets between two icons in the same equivalence class can then be determined from their offsets with respect to their shared cluster representative. The relative orientation of the two icons can then be determined from these offsets by table lookup, while separation comparisons can be done in the r^2 domain, by comparing $\Delta_x^2 + \Delta_y^2$ to the square of the separation constraint.

Test images were generated, consisting of 10 to 90 icons apiece, randomly located on a 1024 by 1024 grid. For each test image, indexes were generated with values of the pruning parameter ϕ of 2, 4, 6, and 8. (Angular unit values from 0 to 255 represent angles from 0° to 360°. Thus, one angular unit represents slightly more than 1.4 degrees. Values of the pruning factor investigated correspond to beam half-widths of approximately 2.8°, 5.6°, 8.5°, and 11.2°, respectively.) Figure 8 shows the (computed) size of an unpruned index, compared to the sizes of the resulting pruned indexes. The same information is shown graphically in Figure 9 using a logarithmic scale.

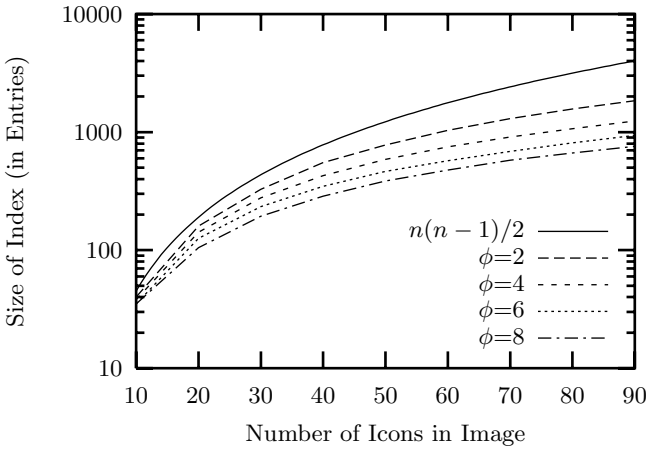


Figure 9: Pruning Efficiency

As the pruning factor ϕ is increased, the size of the index is reduced. The trade-off for this size reduction is that the area of the index space that must be examined is broadened in the θ dimension.

Simulations of search were done for each index produced. A search was run for every icon pair in the image (no simulations of unsuccessful searches were conducted), and the structure of the clusters produced by the Union-Find algorithm was examined.

The simulation showed that for small values of r, small values of ϕ, and sparse data loading, a larger number of smaller clusters were formed. However, as the values of r and ϕ increased, or as the data became more dense, the Union-Find algorithm trended toward production of a singular large cluster.

In a real world implementation the index would be disk resident, thus query costs would be dominated by disk access. For this reason the size of the index and the size of the r-θ space search area were used as a proxy for estimating query costs.

7. SEPARATE CLASS PAIR INDICES AND WILDCARD SEARCHING

While so far the discussion has been limited to the use of a single, unified index, databases storing only a small and limited number k of icon classes might advantageously maintain a separate r-θ index for each of the $k(k+1)/2$ combinations of classes. (This number includes the k homogeneous class pairs A-A, B-B, etc. To achieve the factor of two reduction in the number of indexes, a total order would be induced on the icon classes, as discussed in Section 2.) Either unpruned or pruned indexes might be used, depending on the expected density of icons in that particular database. However, introduction of a single instance of a novel icon class would require immediate instantiation of a relatively large number of new indexes.

In such a multiple-index system, a *wildcard* query (searching for icons belonging to any one of a specific set of icon classes) will generally require more than one of the indexes to be accessed.

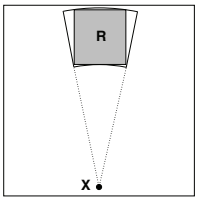


Figure 10: Absolute-position search relative to X

8. DATABASE UPDATE

Adding a completely new database image is straightforward. The pairs for the new image are generated, pruned by the value of ϕ used by that database, and added to the index. Similarly, deletion of an entire image consists simply of removing all the pairs of that image from the index.

Adding an icon to an existing database image can easily be done by generating all the pairs composed of the new icon and the existing icons, then adding into the index only those new pairs for which evidence does not already exist in the index. This is somewhat suboptimal in that one or more of these new pairs may constitute crucial evidence for some of the other pre-existing pairs, allowing them to theoretically be pruned from the index. However, these benefits can be regained by periodically regenerating the index.

Removing an icon cannot be done by simply removing all pairs containing it, as a removed pair may be a crucial part of the evidence for other pairs. Instead, marking the pairs as "logically deleted" without physically deleting them allows their continued participation in the search algorithm but also allows the final filtering pass to remove them from the results. Again, periodically regenerating the index allows the eventual physical deletion of such pairs.

9. NON-POSITION-INDEPENDENT QUERIES

Although the algorithms described above were developed to perform position-independent search, position-dependent queries can be accommodated by adding to the index one artificial icon for each database image, at a point determined by a convention of that particular database (such as "the center" or "the lower left corner"). Query for an icon A in a given area of any image is then done by determining a set of r-θ constraints, relative to the known conventional position of the artificial icon X, that subsumes the given search area. The search is done for an X-A pair with the determined spatial relationship, and the result is filtered to satisfy the original position-dependent query.

In Figure 10, a rectangular search window in normal space can be subsumed into an r-θ query (of type 4B in Figure 2) with respect to artificial icon X. Images found by the algorithms described above can then be filtered to remove any "false positives" generated by this subsumption.

10. NON-SPATIAL QUERIES

An existential query attempts to find database images containing a particular icon A, with no constraint on that icon's position. Such a query could also be satisfied using the artificial X point, by searching for an X-A pair with an unconstrained spatial relationship, but doing so would require that the entire index be retrieved. If a database must frequently support such queries, a separate but parallel inverted-file index might be added.

11. CONCLUDING REMARKS

We have shown how to use a spatially organized index of icon pairs to accelerate search of an image database for icon pairs possessing a desired spatial relationship. The quadratic growth in the storage space required for such an index can be controlled using an algorithm that prunes pairs from the index when their existence can be inferred from remaining unpruned pairs. The trade-off for this pruning is that search must access a larger fraction of the spatially organized index, and may require execution time at worst quadratic in the number of icons per image.

In the two dimensional work described above, the spatial relationship between two icons is characterized as separation (r) plus a single angle (θ). The straightforward extension to three dimensions adds a second angle, characterizing the spatial relationship as a separation plus two angles. The closest familiar analogue might be the "Az-El" (azimuth-elevation) measurement of surveying, in which a direction is characterized as an azimuth (angular bearing in the horizontal plane) and an angular elevation above the horizon, with the range (e.g., distance or separation) supplying the required third coordinate value.

In this case, the shape of the index space would be analogous to a $2\frac{1}{2}$ D map of the earth's surface, or the skin of an orange, with two angular axes, both closed (i.e., a sphere), and with the r dimension extending outward, along the orange skin's thickness. The general pattern for an n dimensional database space would be an index space consisting of a single linear r axis directed "outward" from the surface of a $n-1$ dimensional hypersphere, which would embody the $n-1$ remaining angular dimensions.

It is easier to envision the structure of such an index space than to imagine any real-world application for this technology. Human beings are surprisingly adept at correlating two dimensional maps with physical terrain, given that we did not evolve in an environment where looking down on a terrain from above is an everyday experience. Sadly, this ability does not extend to higher dimensionalities. The only three dimensional analogy that comes to mind is using a database of positions in the solar system (such as space ships, space stations and planets) to find feasible links for an interplanetary relay network, while avoiding radio noise from the galactic core. Higher-order analogs would be even more opaque.

Future research includes extending the analysis from synthetic to real-world datasets, the investigation of algorithms to construct optimal pruned indexes, and algorithms to construct useful pruned indexes in less than $O(n^3)$ time. The space-time tradeoff involved in the employment of a 180° space versus a 360° space is also worthy of investigation.

The existential part of an iconic image database query (i.e., determining the images that contain particular icons, regardless of their spatial arrangement) can be satisfied by a traditional inverted-file index containing icon instances and the images in which they are present. The degree of improvement in real-world iconic image database access efficiency made possible by processing some of a query's spatio-orientational constraints directly in the index is an area open for future quantitative study.

12. REFERENCES

[1] D. Arkoumanis, M. Terrovitis, and E. Stamatogiannakis. Heuristic algorithms for similar configuration retrieval in spatial databases. In *Proceedings of the 2nd Hellenic Conference on Artificial Intelligence (SETN)*, pages 141–152, Thessalonica, Greece, 2002.

[2] Stefano Berretti, Alberto Del Bimbo, and Enrico Vicario. Efficient matching and indexing of graph models in content-based retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1089–1105, October 2001.

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

[4] M. J. Egenhofer. Query processing in spatial-query-by-sketch. *Journal of Visual Languages and Computing*, 8(4):403–424, August 1997.

[5] J. Enderle, M. Hampel, and T. Seidl. Joining interval data in relational databases. In *Proceedings of the ACM SIGMOD Conference*, pages 683–694, Paris, France, June 2004.

[6] C. Faloutsos and W. Rego. Tri-cell—a data structure for spatial objects. *Information Systems*, 14(2):131–139, 1989.

[7] Roop K. Goyal and Max J. Egenhofer. Similarity of cardinal directions. In *Proceedings of the Seventh International Symposium on Spatial and Temporal Databases.*, volume 2121 of *Lecture Notes in Computer Science*, pages 36–55, Los Angeles, California, July 2001.

[8] V. Gudivada. $\Theta\Re$ string: A geometry-based representation for efficient and effective retrieval of images by spatial similarity. *IEEE Transactions of Knowledge and Data Engineering*, 10(3):504–512, May/June 1998.

[9] V. Gudivada and V. Raghavan. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Transactions on Information Systems*, 13(2):115–144, April 1995.

[10] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD Conference*, pages 47–57, Boston, June 1984.

[11] A. Henrich, H.-W. Six, and P. Widmayer. The LSD tree: spatial access to multidimensional point and non-point data. In P. M. G. Apers and G. Wiederhold, editors, *Proceedings of the 15th International Conference on Very Large Databases (VLDB)*, pages 45–53, Amsterdam, The Netherlands, August 1989.

[12] K. Hinrichs and J. Nievergelt. The grid file: a data structure designed to support proximity queries on spatial objects. In M. Nagl and J. Perl, editors, *Proceedings of WG'83, International Workshop on Graphtheoretic Concepts in Computer Science*, pages 100–113, Haus Ohrbeck (near Osnabrück), West Germany, 1983. Trauner Verlag.

[13] A. Holt. Spatial similarity and GIS: the grouping of spatial kinds. In P. A. Whigham, editor, *Eleventh Annual Colloquium of the Spatial Information Research Center (SIRC05)*, pages 241–250, Dunedin, New Zealand, December 1999. University of Otago.

[14] P. W. Huang and Y. R. Jean. Using 2D C^+-strings as spatial knowledge representation for image database systems. *The Journal of the Pattern Recognition Society*, 27(9):1249–1257, September 1994.

[15] U. Lauther. 4-dimensional binary search trees as a means to speed up associative searches in design rule verification of integrated circuits. *Journal of Design Automation and Fault-Tolerant Computing*, 2(3):241–147, July 1978.

[16] S. Y. Lee and F. J. Hsu. 2D C-string: a new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10):1077–1088, October 1990.

[17] D. Papadias, N. Karacapilidis, and D. Arkoumanis. Processing fuzzy spatial queries: A configuration similarity approach. *International Journal of Geographic Information Science*, 13(2):93–128, 1999.

[18] D. Papadias, M. Mantzourogiannis, and I. Ahmad. Fast retrieval of similar configurations. *IEEE Transactions on Multimedia*, 5(2):210–222, June 2003.

[19] J. B. Rosenberg. Geographical data structures compared: a study of data structures supporting region queries. *IEEE Transactions on Computer-Aided Design*, 4(1):53–67, January 1985.

[20] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[21] H. Samet and A. Soffer. MARCO: MAp Retrieval by COntent. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):783–798, August 1996.

[22] B. Seeger and H.-P. Kriegel. Techniques for design and implementation of efficient spatial access methods. In F. Bachillon and D. J. DeWitt, editors, *Proceedings of the 14th International Conference on Very Large Databases (VLDB)*, pages 360–371, Los Angeles, August 1988.

[23] A. P. Sistla, C. Yu, and R. Haddad. Reasoning about spatial relationships in picture retrieval systems. In J. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 570–581, Santiago, Chile, September 1994.

[24] Spiros Skiadopoulos, Christos Giannoukos, Panos Vassiliadis, Timos K. Sellis, and Manolis Koubarakis. Computing and handling cardinal direction information. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT)*, volume 2992 of *Lecture Notes in Computer Science*, pages 329–347, Heraklion, Crete, Greece, March 2004.

[25] A. Soffer and H. Samet. Pictorial query specification for browsing through spatially referenced image databases. *Journal of Visual Languages and Computing*, 9(6):567–596, December 1998. Also an abbreviated version in *Proceedings of the Second International Conference on Visual Information Systems (VISUAL97)*, pages 117–124, San Diego, CA, December 1997.

[26] J.-W. Song, K.-Y. Whang, Y.-K. Lee, M.-J. Lee, and S.-W. Kim. Spatial join processing using corner transformation. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):688–695, July/August 1999.