On Indexing Line Segments

H. V. Jagadish

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

In several image applications, it is necessary to retrieve specific line segments from a potentially very large set. In this paper, we consider the problem of indexing straight line segments to enable efficient retrieval of all line segments that (i) go through a specified point, or (ii) intersect a specified line segment. We propose a data organization, based on the Hough transform, that can be used to solve both retrieval problems efficiently. In addition, the proposed structure can be used for approximate retrievals, finding all line segments that pass close to a specified point. We show, through analysis and experiment, that the proposed technique always does as well as or better than retrieval based on minimum bounding rectangles or line segment end-points.

1. INTRODUCTION

Interest in spatial databases has burgeoned of late, as it has become practical to construct large databases that comprise spatial rather than alphanumeric data. Several excellent approaches have been devised for indexing points in a multi-dimensional space, and also for indexing rectangular regions. See, for example, [2, 9, 11, 13, 15]

Given an arbitrarily shaped object, the typical strategy is to create a rectangular bounding region (or a polyhedral bounding region) of the object, and then to index this bounding region. (Sometimes multiple smaller rectangles may be used to cover the object). The disadvantage is that the bounding region is necessarily larger than the object itself so that some "false hits" will be generated. This penalty may be acceptable if the bounding region is only a little bit larger than the object that it bounds. However, in the case of a line segment, the bounding region typically has an area (or volume) much greater than the line itself.

So far, not much attention has been focussed on devising index structures for line segments in particular. The coordinates of the end points of the line segment can be used for indexing. Thus a line segment in m-dimensional space can be represented as a point in a $2m$-dimensional space, and any well-known index structure can be built over these points. For example, the grid-file has been suggested in [4] and the quad tree in [14]. Such an index method is of value only when the retrieval is based upon the coordinates of the end-points of the line. The index provides little advantage for a retrieval based on some intermediate point on the line.

A better option is to divide the space into cells, and store with each cell a list of line segments that intersect it. Several different techniques have been proposed to divide the space into cells. For example, edge Excell [17] and the PMR quadtree [8]. A drawback of these techniques is that information regarding each line segment has to be stored several times, once for each cell that the line segment intersects. Two line segments can intersect only if they pass through at least one cell in common. Therefore, the smaller the cell size the greater the selectivity, but also the greater the storage cost due to information duplication. (Consequently much effort has been devoted to sizing the cells adaptively).

In this paper we propose an index structure for line segments, which requires storing a line segment only once, and using which it is possible to perform the following retrievals efficiently:

i. Find all line segments that pass through a specified point,

ii. Find all line segments that lie in the vicinity of a specified point, and

iii. Find all line segments that intersect a specified line segment.

Proceedings of the 16th VLDB Conference
Brisbane, Australia 1990

In addition, the following simpler retrievals fall out as easy by-products of the index structure constructed:

i. Find all line segments that are parallel or perpendicular to a specified line,

ii. Find all line segments that are coincident with a specified line,

iii. Find all line segments that intersect a specified (infinite) line,

iv. Find all line segments that include a specified set of points, and

v. Find all line segments that have specified end-points.

We begin by presenting in Section 2 some motivation for studying the line segment indexing problem. Background material covered in Section 3 includes a quick tutorial on the *Hough Transform*, a simple mathematical operation that is central to the data structure proposed in this paper.

Section 4 is the heart of the paper, in which we present our proposed data structure and show how each of the types of queries listed above can be handled. In Section 5, we analytically compare our indexing to the traditional bounding rectangle method. We confirm our analysis with simulation experiments described in Section 6.

Sections 2-6 deal only with line segments in a two-dimensional plane. In Section 7, we show how these ideas can be extended to line segments in a multi-dimensional space. We wrap up with some final comments in Section 8.

2. MOTIVATION

In machine vision, line segments are found from edges of objects, and these line segments are then used for further procesing. For example, to facilitate higher level recognition of a pattern or object, one has to determine mutual relationships between pairs (or sets) of these line segments. In this context, one may wish to find all line segments that pass through the end-point of a specified line segment. One may also wish to determine all line segments that intersect a given line segment. These would be the line segments with which the given line segment is likely to have an interaction or relationship. This information, when passed to a higher level in the recognition process, can be used to find known groupings of line segments.

For example, an optical character recognition system can be programmed to recognize as the Hindu numeral 4, any set of three line segments where segments A & B share an end-point and are at an angle of 45° to 90°, segment B is within 15° of the horizontal, and segment C intersects segment B at an angle of 60° to 120°. Such a system would recognize as the numeral 4, any of the line segment groups in Fig. 1. (In a real system, the rules would be similar in spirit to the ones given above, but could be far more complex).

The basic idea described above has been used in many different applications. For example, in [1], line segments are found in an image and matched against a reference, in the context of scene understanding for robots with machine vision. In structured light machine vision systems, stripes of lights are projected on an object for three-dimensional recognition [18]. The edges of these stripes are line segments whose intersection with object features has to be determined. In [10] line segments are fit to the contours of a fingerprint, and used for pattern recognition. Similarly, for recognition of roads and boundaries in geographical maps, satellite images, and aerial images, line segments are the common primitive [7].

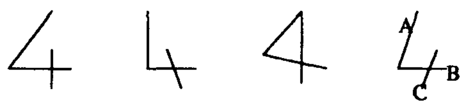

Figure 1. A few different ways of writing the numeral 4, which can be recognized as a set of three intersecting line segments: A, B, and C

Line segments are useful in a completely different context as well. Electrical connectivity desired on a Printed Circuit Board is obtained by the use of piecewise linear metallic traces. As boards have grown larger and more complex, these traces are "laid out" by computer, rather than by hand. In an involved design, an important integrity check is that two traces on one layer should intersect only if they belong to the same electrical net. This problem boils down to finding all pairs of intersecting line segments in a given set.

Similarly, when creating a "via", or a hole from one layer to another in a PC board, one has to ensure that the hole does not go through any traces in any of the intervening layers. Treating the hole as a point, this problem translates into ascertaining that no line segments go through the specified point in each of the intervening layers.

Thus we see that in diverse applications, it is important to be able to determine efficiently the set of all line segments that either go through a specified point or intersect a specified line segment. It is these problems that we address in this paper.

3. PRELIMINARIES

The Hough transform [5] was developed as an aid to pattern recognition, and is widely used today. The basic idea in the transform is that each (infinite) line (in a two-dimensional plane) can in general be written in the form $y = mx + b$. In other words, it is determined uniquely given its slope, m, and its intercept on the Y-axis, b. So, in a *transform space*, in which slope is plotted along one axis and intercept along the other, every point uniquely determines and is uniquely determined by a line in the regular space. Thus the Hough transform provides a one-to-one mapping of lines in the original space to points in the transform space.

Observe that vertical lines are not treated properly. A vertical line has an "infinite" slope, and corresponds to a point "at infinity" in the transform space. In fact, we would like to avoid lines with a large slope since they cause an explosion of the ranges of values of transform space coordinates. We observe that the line, $y = mx + b$, could also be written as $x = ny + c$, where $n = 1/m$ and $c = -b/m$. In other words, only one of n and m can have an absolute value greater than one for any given line, with $n = m = \pm 1$ for lines at a $\pm 45°$ angle. Let us call the regular Hough transform, defined in the previous paragraph, the *Hough-X* transform. We can then define a corresponding *Hough-Y* transform, into the n,c transform space, rather than the m,b transform space. For every line, note that at least one of the Hough-Y and Hough-X transforms is well-defined.

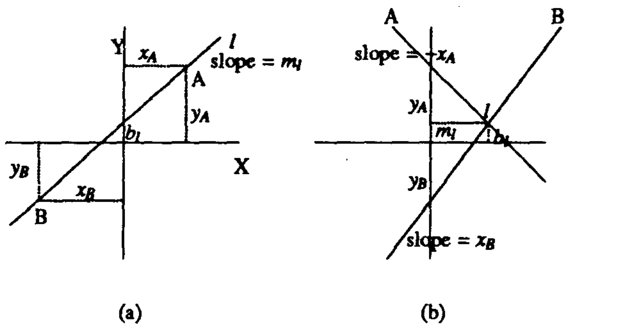

Figure 2. (a) A line in a plane, and two points on it. (b) The Hough-X transform of the line is a point, and of the points is a pair of lines.

Now consider a point in the original space. Every pair of lines determines a point at their intersection[1]. But the pair of lines correspond to a pair of points in the transform space, and these determine a line in the transform space. Thus, a point in the original space corresponds to a line in the transform space. Mathematically, let the point be (x_0,y_0). Every line that goes through it must have a slope and intercept such that $y_0 = mx_0 + b$. This equation can be rearranged and written, $b = -x_0m + y_0$, which represents a straight line in the m,b transform space with a slope of $-x_0$ and an intercept of y_0. Similarly, the Hough-Y transform of the point can be written, $c = -y_0n + x_0$, which is a straight line with a slope of $-y_0$ and an intercept of x_0. Note that for every point (with bounded coordinates) both the Hough-X and the Hough-Y transforms are defined, and that both the slope and the intercept of the resulting line are bounded. See Fig. 2.

4. PROPOSED TECHNIQUE

4.1 The Data Structure

Given a database, **D**, of line segments in a (two-dimensional) plane, partition it into two sets. In one set, $\mathbf{D_X}$, place all line segments that have slopes in the range [-1,+1]. In the other set, $\mathbf{D_Y}$, place all line segments with slopes outside this range. Extend each line segment in $\mathbf{D_X}$ to an infinite line, and then obtain the Hough-X transform of the line. Thus obtain one point in the transform space corresponding to each line segment in $\mathbf{D_X}$.

1. A pair of parallel lines determine a point at infinity.

Suppose that the X coordinate in $\mathbf{D_X}$ ranges from x_{min} to x_{max}, and the Y coordinate ranges from y_{min} to y_{max}. Then the m coordinate in the transform space ranges from -1 to $+1$, and the b coordinate at most from $y_{min} - \max(|x_{min}|, |x_{max}|)$ to $y_{max} + \max(|x_{min}|, |x_{max}|)$. In other words, the ranges of values in the transform space are bounded, and are comparable to the ranges in the regular space. The area bounded by these ranges in transform space will be called the *region of interest*.

Four numbers determine a line segment. Traditionally, these could be the X and Y coordinates of both end-points. Instead, for each line segment in $\mathbf{D_X}$, the four values are stored along three different attribute axes:

i. The slope of the line segment. That is, the m coordinate.

ii. The Y-intercept of the infinite extension of the line segment. That is, the b coordinate.

iii. The projection on the X axis of the line segment. That is, the range x_{min} to x_{max}.

Similarly, for each line segment in $\mathbf{D_Y}$, values are stored along the three complementary attribute axes.

i. The inverse slope of the line segment. That is, the n coordinate.

ii. The X-intercept of the infinite extension of the line segment. That is, the c coordinate.

iii. The projection on the Y axis of the line segment. That is, the range y_{min} to y_{max}.

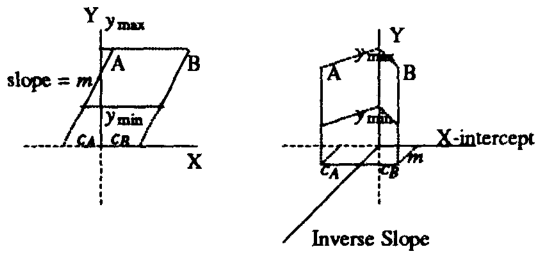

Figure 3. Two parallel line segments in $\mathbf{D_Y}$ shown in the X-Y plane, and after transformation for our data structure, in three-dimensional space.

Thus an arbitrarily oriented line segment in a plane is transformed into a vertical line segment in three-dimensions. See Fig. 3. Any standard multi-dimensional range indexing technique, such as grid file [9], R-tree [2], R^+-tree [16], P-tree [6], *LSD*-tree [3], or z-ordering [11], can be used for each of $\mathbf{D_Y}$ and $\mathbf{D_X}$. If necessary, the range on X (or Y) can be mapped to two coordinates on different axes, giving a point in four-dimensional space.

4.2 Some Simple Queries

4.2.1 Coincident with Specified Line

Obtain the slope of the specified line. If the absolute value of the slope is less than or equal to 1, then obtain the Y-intercept as well. The slope and the Y-intercept provide a selection on two attributes of elements in the set $\mathbf{D_X}$. There is no selection on the range attribute. One can use any standard partial match technique to retrieve all line segments in the database that are coincident with a specified line. Observe that there is no need to examine D_Y at all. On the other hand, if the absolute value of the slope is greater than 1, use the inverse of the slope and the X-intercept to provide a selection on two attributes in D_Y.

4.2.2 Parallel or Perpendicular to a Specified Line

Obtain the slope of the line. Depending on whether its absolute value is greater than one, use it or its inverse to perform a single attribute selection on either D_Y or D_X. No selection is specified on the intercept or range attributes. Thus obtain all line segments parallel to a specified line.

To find all line segments perpendicular to a specified line, take the inverse of the slope of the specified line, change its sign, and then proceed as above. In effect, find a line perpendicular to the specified line, and then find line segments parallel to this new line. The same idea can be used to retrieve all line segments at a specified angle (or range of angles), either absolute or relative to a given line.

4.2.3 Specified End Points

Using the end points specified, one can compute the slope (or inverse-slope), the intercept (X or Y as required), and the range (Y or X as required). One thus has a selection specified on all attributes, and can use this to retrieve the relevant line segment(s) from D_Y or D_X as the case may be.

4.2.4 Specified Intermediate Points

As before, if two intermediate points are specified, they determine an (infinite) line, which has a specific slope and intercept. We then have a retrieval with a completely specified selection on two attributes, and a requirement on the other attribute that the retrieved items include the specified X (or Y) range obtained from the two points.

If more than two points are specified, first check to see that they are collinear. If so, use the two outermost points and apply the procedure above. If not, there cannot be a line segment that passes through them all.

4.3 One Specified Point

Theorem 1

A line segment passes through a specified point if and only if it satisfies the following two conditions:

i. The X (or Y) range includes the specified point. That is, given a query point (x_0,y_0), the line segment must have $x_{min} \le x_0 \le x_{max}$.
ii. The point corresponding to the line segment in transform space lies on the line in transform space corresponding to the specified point. That is, $b = -x_0m + y_0$, where b is the Y-intercept, and m is the slope of the line segment, for a line segment in D_X (or $c = -y_0n + x_0$, for a line segment in D_Y).

Proof:
The only if part is straightforward. If only one point is specified, a line segment of any arbitrary slope and intercept could pass through this point. However, we know that the point corresponds to a line in the Hough-X transform space and a line in the Hough-Y transform space. Any line segments that pass through this point must correspond to a point on one of these lines in the appropriate transform space. Moreover, a line segment can pass through a point only if the point lies within the X-range (and Y-range) of the line segment.

The if part is only a little more difficult. Since condition (ii) is satisfied, the infinite extension of the line segment includes the specified point. However, all points in the infinite extension of the line segment will have coordinates outside the X and Y ranges of the line segment. That is, the infinite extension of the given line segment contains no points in the range $[X_{min}, X_{max}]$ that are not in the line segment itself. Similarly for the Y coordinate. But, from condition (i), the specified point of interest has an X coordinate in $[X_{min}, X_{max}]$ (or a Y coordinate in $[Y_{min}, Y_{max}]$). Therefore if the specified point is included in the infinite extension of a line segment, then it must be included in the line segment itself. □

Thus we have transformed the retrieval problem into one of selecting vertical line segments that intersect a specified query region in three-dimensional space. The query region is planar, obtained as a product of a line segment parallel to one (the range) axis, and a line in the plane defined by the other two axes. By splitting the data set in two, and generating two separate queries, one for each transform space, we have ensured that the line in the plane is bounded, giving us a finite query region. This region can be approximated, if necessary, by multiple small rectangles that cover the region, so that standard rectangle retrieval techniques can be used thereafter. Observe that it is only the query region that is being approximated, and not the data items. Therefore, there is no increase in storage required for data or indexing, even if many rectangles are used to obtain a very good approximation of the search region.

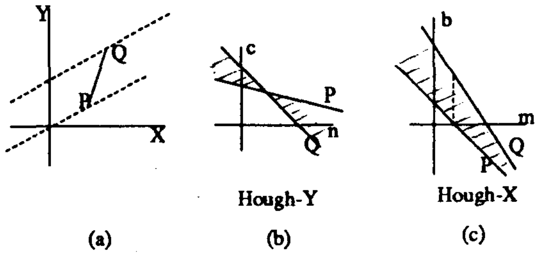

Figure 4. A line segment in the X-Y plane, and the Hough-Y and Hough-X transforms of its end-points. Since the line segment has a slope greater then 1, the transformed end-points meet only outside the area of interest in the Hough-X transform space. The region between the two lines is shown hatched. The dashed line in Fig. 4c corresponds to lines parallel to and between the dashed lines in Fig. 4a.

4.4 Intersecting a Specified Line Segment

4.4.1 A Simple Technique

Take the two end-points of the specified line segment and apply the Hough-X transform. We thus obtain two lines in X-transform space. If the absolute value of the slope of the given line segment is no greater than 1, then these two lines will intersect within the area of interest in the X-transform space. If not, then the two lines will not intersect in the area of interest. Similarly, the end-points of the given line segment can be used to create two lines in the Y-transform space. See Fig. 4.

Each intermediate point on the line segment is represented by a line in each transform space that lies "between" the lines corresponding to the two end-points. Any (infinite) line that intersects the given line segment must pass through an intermediate point, and hence must lie "between" the two lines in the appropriate transform space. Therefore we need consider only those line segments whose infinite extensions transform to points that lie "between" the pair of lines obtained by transforming the end-points of the specified line segment. This notion of betweenness is illustrated in Fig. 4 and formally defined below.

Call the two lines, P and Q. Consider first the case where the two lines do not intersect within the region of interest. Line P divides the region of interest into two regions, H_1 and H_2. Since line Q does not intersect line P within the region of interest, it passes through only one of the two regions. Without loss of generality assume that is region H_1. Similarly, line Q divides the region in two, and line P passes through only one of these two regions, say, H_3. Then $H_1 \cap H_3$ is the desired region *between* the pair of lines P and Q.

Now consider the case where the two lines intersect at a point within the region of interest. The lines intersect in a point that is the Hough transform of the given line segment. Clearly, no line with the same slope can intersect this line segment. That is, no point in transform space with the same m (or n) coordinate can be of interest. Two intersecting lines divide the region of interest into 4 parts. Two of these parts are eliminated by the above rule. The remaining two parts are the ones that we consider to lie *between* the two lines. Another way of looking at it is that for any given slope, the two lines determine the extreme values of the intercepts that a line can have and still intersect the line segment. At the intersection point of these two lines, the intercept is uniquely determined. At all other points, it is a range.

In addition to the restriction on slope and intercept discussed above, two line segments cannot intersect unless their X (and Y) ranges intersect. Thus our search region in the three-dimensional transform space is given by the area between the lines in the m,b (or n,c) plane, multiplied by the height of the X (or Y) range along the third orthogonal axis. (See the Appendix for the mathematics). The retrieval is of all (vertical) line segments that intersect this region in three-dimensional transform space. As before, two retrievals are required, one for the line segments in $\mathbf{D_X}$, and another for the line segments in $\mathbf{D_Y}$.

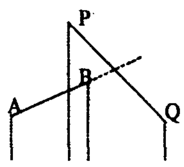

Figure 5. The extension of line segment AB intersects query segment PQ, and their X-ranges intersect, but the line segments do not intersect

4.4.2 An Exact Technique

If we perform the retrieval suggested above, we will still get some line segments that do not intersect the specified line segment. See Fig. 5. The problem is that even though we have ensured that the infinite extension of a retrieved line segment, such as AB in Fig. 5, does intersect the specified query line segment, PQ, the range condition we have set is too weak. Consequently, some excess line segments, such as AB, are retrieved.

Consider the region between the lines corresponding to the end-points of the query segment in the transform plane. Each point in this region corresponds to an (infinite) line that intersects the given line segment. This intersection is at a point, which has a single X-coordinate (and Y-coordinate). A line segment with the corresponding slope and intercept will intersect the query segment iff its X range includes this intersection X-coordinate (or Y range includes the intersection Y-coordinate). Therefore, rather than specifying a range of value along the X (or Y) axis, we can actually compute and specify a unique X (or Y) coordinate value for each slope-intercept pair. Thus, the query region becomes a two-dimensional manifold in the three dimensional space.

A line segment intersects a specified query line segment if and only if its transform intersects the manifold obtained as above.

What does this manifold look like? For every value of the slope, there is a range of values of intercepts, and a corresponding range of values of intersection points for line segments that intersect the query segment. For example, the dashed line in Fig. 4c corresponds to a set of parallel lines between the two dashed lines in Fig. 4a. The intersection point is simply P, at the bottom end of the dashed line in Fig. 4c, with a corresponding X coordinate. At the top end of the dashed line, the intersection point is Q, with a corresponding X coordinate. Between these two ends, as the intercept value is changed, the X coordinate of the intersection point also changes — linearly. The equation is derived in the Appendix. A similar argument applies for every choice of slope value. The X value for the intersection point, is constant along the P line, being the coordinate of the point P itself. Similarly, it is constant along the Q line. Imagine a scroll, or a window blind, with a stiff top and bottom. Pull the bottom out a little so that it is not in alignment with the top. This is roughly what our manifold looks like.

Turn now to the case of Fig. 4b. The same arguments applies as above. The only difference is that as the slope gets closer to the slope of the query segment, the range of permissible intercept values gets smaller and smaller, while the range of Y (or X) coordinates is not altered. When the slope exactly equals the slope of the query segment, only one intercept value is permitted, and that is the intercept value of the query segment itself. For this choice of slope and intercept, the entire range of Y (or X) coordinate values of the query segment are acceptable. You could imagine twisting the bottom end of the scroll around, while still keeping both top and bottom perfectly horizontal, so that in the middle of the twist there is a point where the scroll is vertical.

4.4.3 Intersecting a Specified (Infinite) Line

Lines with every slope and intercept will intersect any given infinite line query object. The only interesting selection we can apply is in terms of the range coordinate. The intersection point of any line (specified in terms of its slope and intercept), with the query line, can be determined uniquely[2]. We thus obtain a two-dimensional manifold in our three-dimensional attribute space. We must retrieve the transformed vertical line segments that intersect this manifold, in response to the query.

Another way of thinking about this problem is to take the end points of a query line segment, move them infinitely apart, and solve the line segment intersection problem above. The manifolds described above would still be of the same basic form, but would simply get stretched (in the intercept and range dimensions). Since the given database has some finite bounds on the coordinates, and hence intercepts of the line segments in it, we need perform the retrieval only on a finite portion of the infinitely stretched manifold.

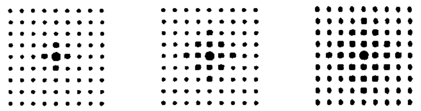

Figure 6. A pixel and its neighbors, 1, 2, and 3 pixels away

4.5 Close to a Specified Point

Due to noise or error in the image creation, it may often be the case that a line that is "supposed" to go through a point actually does not do so. Similarly, in a printed circuit board layout, design rules may require a minimum gap between a trace and a via, as a safety margin. When queried for line segments that pass through a specified point, one may therefore wish to include line segments that "nearly" go through the point. From a purely geometric viewpoint, we may be interested in line segments that pass a distance no greater than δ away from a given point, essentially requiring the retrieval of all line segments that pass through a circular region. However, in the digital domain, one has discrete pixels, and error is more appropriately measured in number of pixels away. Using the standard convention of each pixel having four neighbors, Fig. 6 shows all pixels upto one pixel away, upto two pixels away, and upto three pixels away from a specified point (pixel). Observe that each of these k-pixel neighborhoods, for any positive integer k, is a square rotated through 45°. Therefore, the problem of interest is to find all line segments that pass through such a rotated square.

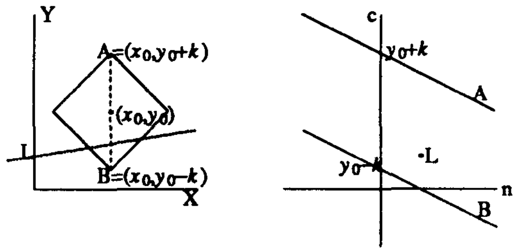

Figure 7. A rotated square query region and a line intersecting it are shown in the original plane and in the Hough-Y transform space. Observe that the line L is in D_X, and intersects the vertical disgonal of the square (shown dashed), but does not intersect the horizontal diagonal.

Consider lines with absolute value of slope no greater than 1. These are lines that are closer to the horizontal than 45°. If any such line intersects a rotated square, then it passes through the vertical "diagonal" of the square. See Fig. 7. Thus the problem reduces to finding lines that intersect this vertical line segment. Using the technique of Sec. 4.4, we observe that the Hough-X transform of the end-points of the vertical line segment are a pair of parallel lines. (There is no need to take the Hough-Y transform of this line segment since we are currently considering only line segments with absolute slopes less than one). A line segment in D_X intersects a given rotated square only if the Hough-X transform of its infinite extension lies in the region between the two parallel lines obtained by a Hough-X transform of the top and bottom vertices of the rotated square. Moreover, a line segment intersects a given rotated square only if its X-range intersects the X-range of the rotated square. As in the previous sections, all line segments in D_X that intersect the given rotated square can be retrieved by means of a selection on all attributes that arises from these conditions. The selected region is a parallelopiped.

Once more, using the simple range selection described in the preceding paragraph is much too generous. We observe, as in Section 4.4.2, that a line segment passing through the rotated square has a specified range of X coordinate values in the intersection, as a function of its slope and intercept. This range of coordinate values can be described as a piecewise linear function, continuous but not differentiable each time a vertex of the rotated square is crossed. The three-dimensional region thus obtained provides a perfect selection of line segments that intersect the (interior of) the given rotated square query region. See the Appendix for the mathematics.

Similar arguments can be applied to line segments in D_Y, using the left and right end vertices of the rotated square and its "horizontal" diagonal.

2. With the case of slope and intercept equal to the query line being a point of singularity.

5. PERFORMANCE ANALYSIS

There are two previous proposals against which we must compare ours. One is to use rectangular bounding boxes for each data line segment, and for the query regions, and then to use an index structure on these bounding boxes, such as the R-tree. The other is to use the coordinates of the end-points of the line segments to map the line segment to a point in a four-dimensional space, and then to index these points. One could then use an index structure such as the grid file.

It is easy to see that the selectivity obtained by these two methods is identical. Since bounding rectangles in two dimensions are easier to visualize than equivalent structures in four dimensions, we compare the proposals of this paper against bounding rectangles in what follows.

Each line segment is stored and indexed by means of four parameter values, both in our proposal and in previous proposals. To this extent, one should not expect the storage required for the data, or for indexing, to be very much different for our technique and its competition.

On the other hand, whereas traditional (end-point or rectangular bounding region) techniques retrieve more items than specified in a selection, we are able to perform a retrieval of exactly the set of selected items. Through analysis in this section, we place a quantitative measure on this improvement.

Performing an exact retrieval using our technique requires that we use an index structure that can handle a query region that is a polyhedron. Most indexing techniques today permit only rectangles or, at the most, polyhedra with specified face angles [6]. To match available index structures, our query polyhedron will have to be approximated by multiple rectangular regions that together cover the query region. Retrievals are then performed on these rectangular regions. Observe that no additional storage is involved, and that the data and index structures are not altered due to the query. The query itself is simply divided into smaller queries that can be answered efficiently. See [12] for an excellent study of a similar problem in a different context.

Some improvement can be obtained in the traditional techniques also, if the query region is thus divided up into smaller rectangles. However, this improvement is small. The major hurdle is that the data-set still comprises arbitrarily oriented line segments each of which has a single bounding rectangle. Selectivity can be improved by using multiple bounding rectangles in the data-set as well, but this results in an increased storage and indexing cost, and is not considered here.

This section is organized as follows. In Section 5.1, we discuss previous proposals that compete with ours. In Section 5.2, we describe the analytic technique used in the rest of this section. Having established these preliminaries, we discuss the point (and region) intersection problem in Section 5.3, and the line segment intersection problem in Section 5.4. In both cases, we obtain solutions for the performance of four techniques:

i. Bounding rectangles for data and for query. Call this technique B (for Both Bounded).

ii. Bounding rectangle for data items only, but with an exact query region. Call this technique D (for only Data items bounded).

iii. Our proposal for an exactly specified search region. Call this technique E (for Exact).

iv. Our proposal, with the range axis specified orthogonally. Call this technique S (for transform space index with Simplified range condition).

5.1 Analytic Technique

Consider a line segment of a known length d, at a random angle θ from the horizontal. Without loss of generality, we shall assume that θ is a positive angle: the arguments are symmetric for negative values of θ. We shall determine what the position of the lower left end point[3] of the segment must be for it to intersect the specified object, where this object may be a region, a point or another line segment.

Assume that the lower left end-points of the line segments in the database are uniformly distributed. This is likely to be true neglecting boundary effects, that is, if the length of a line segment is a small fraction of the range of values in the database. Then the area of the region in which the lower-left end-point must lie, for the line segment to intersect the given object, is a measure of the selectivity of the given query. (The fraction of line segments in the answer set is the area of this region normalized by the area of the entire database).

Similar regions can be computed for the positions of the lower left end-points that cause a line segment to be retrieved in response to the query, for any particular indexing method used. The areas of these regions are again proportional to the selectivity of the corresponding indexing method, and can be used to compute the effectiveness of different index strategies.

The results obtained can be integrated over the possible values of the angle and length of the lines segments, using any specified (possibly non-uniform) distributions, if desired.

5.2 Intersecting a Region

Consider a rotated square region of side s. The bounding rectangle of this region is a square of size $\sqrt{2}s$, as shown in Fig. 8a. A line segment of length d at an angle θ has a bounding rectangle that is $d\cos\theta$ wide and $d\sin\theta$ high, as shown in Fig. 8b. Such a line segment will be retrieved as potentially intersecting the square, using the bounding rectangles intersection method, if its lower left end point is anywhere in the region shown in Fig. 8c. The area of this region is $(\sqrt{2}s + d\cos\theta) \times (\sqrt{2}s + d\sin\theta) = 2s^2 + d^2\sin\theta\cos\theta + \sqrt{2}sd\cos\theta + \sqrt{2}sd\sin\theta$. If the rotated square region is represented exactly and not approximated, we save s^2, as can be seen from Fig. 8d.

3. The left end-point if the line segment is horizontal, and the lower end point if it is vertical. In all other cases one end point is both lower than and to the left of the other.

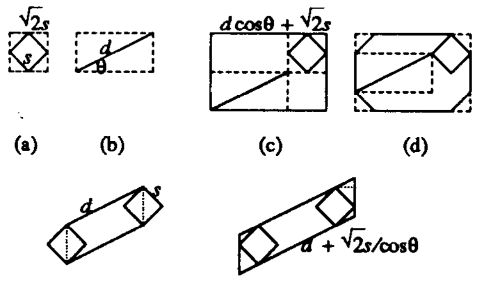

Figure 8. The region of possible locations for the lower left end-point of a line segment intersecting a rotated square query region. (a) The query region and its bounding rectangle, (b) A typical line segment and its bounding rectangle, (c) The region found by bounding rectangles on query and data, (d) The region found by bounding rectangles on data, but using the query exactly, (e) The region found by the exact retrieval technique using our data structure, and (f) The region found by the simple retrieval technique using our data structure.

The actual set of lower left end-point values for a line segment that intersects the region can be obtained by "sweeping" the region to the left and down at an angle θ for a distance d. This region is the one that our technique would find and is shown in Fig. 8e. The area of this region is $s^2 + \sqrt{2}sd\cos\theta$, which is strictly less than the area obtained from the bounding rectangles method above. If we use the simpler range computation technique with our data structure, then we pay an area penalty of s^2 as shown in Fig. 8f: there is a single larger parallelogram and its area is $\sqrt{2}s \times (d + \sqrt{2}s/\cos\theta) \times \cos\theta$. Fig. 8 considers the case where $\theta < 45^\circ$, so that the line segment belongs to D_X. For line segments in D_Y, the figures would look essentially similar, and the computations are identical. The only exception is Fig. 8f, where the larger parallelogram has two of its sides horizontal rather than vertical, since a Y range condition is used isntead of an X range condition. The extra area due to the simplified range condition is still s^2.

Assuming that the angle θ is uniformly distributed, we can integrate over it to obtain an expected area for each method as shown in the table below. If d is small compared to s, then the savings from our method is small. Methods that bound the query region exactly win, by a factor of roughly two, over methods that using a bounding rectangle for the query region. However, if s is small compared to d, then the savings from our method could be a very large factor. For the applications discussed in Section 2, s is expected to be small, since we typically want to find line segments that pass close to a point of interest.

In fact, for the point intersection problem, s goes to zero, giving us a theoretically unbounded benefit. However, considering a discrete image, a point is one pixel. So setting $s = 1$, we obtain the result that our method outperforms bounding rectangles by a factor of approximately d, the expected length of a line segment in the databases, measured in pixels.

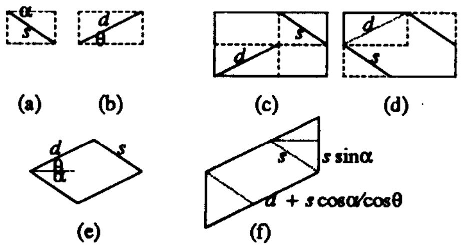

Figure 9. The region of possible locations for the lower left end-point of a line segment intersecting another line segment. (a) The query region and its bounding rectangle, (b) A typical line segment and its bounding rectangle, (c) The region found by bounding rectangles on query and data, (d) The region found by bounding rectangles on data, but using the query exactly, (e) The region found by the exact retrieval technique using our data structure, and (f) The region found by the simple retrieval technique using our data structure.

5.3 Intersecting a Line Segment

Consider a line segment L, of length s at an angle α, as shown in Fig. 9a. Now we can no longer assume α to be a positive angle without loss of generality. So we let α range between $-\pi/2$ and $\pi/2$. The bounding rectangle of this line segment is of width $s\cos|\alpha|$, and height $s\sin|\alpha|$. As before, the bounding rectangle of L intersects with the bounding rectangle of a candidate line segment of length d and positive angle θ, if and only if the lower left end point of the latter lies in a region of area $(d\cos\theta + s\cos|\alpha|) \times (d\sin\theta + s\sin|\alpha|)$. See Figs. 9b and 9c. If the query line segment is not approximated by a bounding rectangle, then the search region, as shown in Fig. 9d, saves two triangular regions at the top-right and bottom-left corners with a combined area of $s\cos|\alpha| \times s\sin|\alpha|$, for a total area of $d^2\cos\theta\sin\theta + sd\sin\theta\cos|\alpha| + sd\cos\theta\sin|\alpha|$.

The actual region in which the lower left end-point of a line segment intersecting L may lie can be obtained once more by "sweeping" L to the left and down at an angle θ for a distance d. The result is a parallelogram with sides of length s and d, as shown in Fig. 9e. The area is $sd\sin(|\theta-\alpha|)$. A larger parallelogram is obtained, as shown in Fig. 9f, if we use our proposal with the simplified range condition. The area is a little messy to compute and depends on the relative values of α and θ. For the specific case shown in Fig. 9f, with $\alpha<0<\theta$, the area is
$(d + s\cos|\alpha|/\cos\theta) \times (s\sin|\alpha| + s\cos|\alpha|\tan\theta) \times \cos\theta$.
The general solution can be shown, through analyzing such cases, to be
$(sd + s^2 \frac{\cos|\alpha|}{\cos\theta})\sin(|\theta-\alpha|)$, provided that the data line segment belongs to D_X. The solution for data line segments in D_Y can be derived:
$(sd + s^2 \frac{\sin|\alpha|}{\sin\theta})\sin(|\theta-\alpha|)$.

Technique	Region Query	Line Segment Query
B (Bounding Rectangles)	$2s^2 + 4\sqrt{2}sd/\pi + 2d^2/\pi$	$d^2/\pi + s^2/\pi + 8sd/\pi^2$
D (Data Bounding Rectangle)	$s^2 + 4\sqrt{2}sd/\pi + 2d^2/\pi$	$d^2/\pi + 8sd/\pi^2$
E (Our Exact Proposal)	$s^2 + 2\sqrt{2}sd/\pi$	$2sd/\pi$
S (Our Simplified Proposal)	$2s^2 + 2\sqrt{2}sd/\pi$	$2sd/\pi + s^2(\frac{1}{\pi} + \frac{1}{12})$

TABLE . Relative Performance of the Algorithms

Once more, we can assume that the angles θ and α are uniformly distributed, and integrate over them to obtain an expected area for each method as shown in the table. As in the case of region queries, our proposal always wins, but wins by the largest margin when d is much larger than s, that is, when the query line segment is much smaller than most line segments in the database. The integral for the case of our simplified proposal is messy to evaluate exactly. Instead, the value in the table is an upper bound for the integral. The actual value will be slightly smaller.

6. EXPERIMENTAL EVALUATION

To confirm the predictions from the analysis in the previous section, we ran some simulation experiments. All four techniques discussed in the previous section were evaluated against both line segment queries and region queries. Four curves marked 'B', 'D', 'S', and 'E', respectively plot the performance of these techniques in Fig. 10 and 11. 'B' for Both query and data item are approximated by Bounding rectangles; 'D' for only the Data item has a bounding rectangle, 'S' for our proposal with the Simple range condition, and 'E' for our proposal with the Exact range condition.

The appropriate performance metric to compare the four techniques should be the number of disk blocks fetched in response to different queries. However, the number of disk blocks fetched depends on the exact physical clustering strategy used, a subject that is outside the scope of this current paper. So we use the number of data items fetched as the measure of performance instead.

Ideally, one should run experiments against the actual database, or at least a synthetic database with the expected statistics of the actual. A characterization of the parameters of a line segment database, and performance studies as these parameters are varied, are a current topic of research. In the absence of such characterization, all experiments were run on a synthetic database of 10,000 line segments that were randomly generated by choosing each end point coordinate from a uniform distribution over a range, [-1000,+1000].

Query line segments were randomly generated, with the length controlled as a parameter. Fig. 11 shows a plot of the relative number of line segments fetched for a query line segment as the length of the query segment is varied. Each point in the graph is obtained by computing the number of items fetched for 100 different random query segments. The total number of items retrieved is plotted as a multiple of the number of items that actually satisfied the specified selections. The selectivity of the query, in terms of the actual fraction of

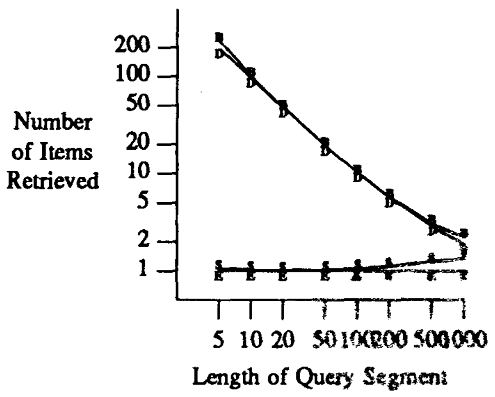

Figure 10. Relative number of line segments retrieved by each technique in response to a line segment intersection query

line segments that should be retrieved (and are retrieved upon exact selection) varies from less than 0.05% for very short query segments to over 10% for very long ones.

For small query segments, we see that our proposals, S & E, perform a couple of orders of magnitude better than the competition. Note the log scale of the plot. As the query segment gets larger, bounding rectangles start performing relatively much better. However, even for extremely large query segments, the exact technique, E, is still ahead by a factor greater than 2.

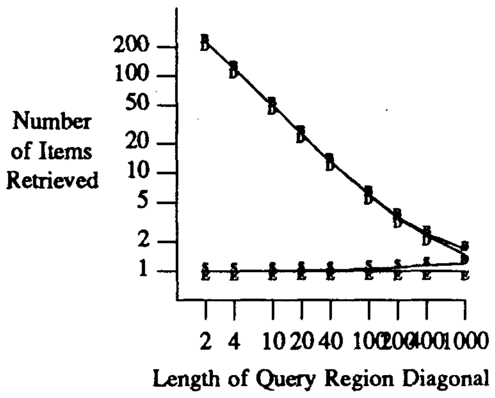

Figure 11. Relative number of line segments retrieved by each technique in response to a rotated square region intersection query

Fig. 12 shows the four techniques used against a rotated square query region, as the size of this square is changed. Once more, each point is obtained from 100 randomly centered rotated square query regions of the specified size, and plotted as a ratio. The selectivity varies from less than 0.05% for a very small rotated square (effectively a point) to over 30% for huge rotated squares that encompass a large fraction of the database. The trends are very similar to those in Fig. 11. Our proposals win handsomely for small query regions, and win by a small margin for large query regions. The simplified range criterion technique performs almost as well as the exact technique except for very large query regions.

Overall, we find that our proposals greatly outperformed the current techniques in the selectivity provided, as measured in our simulations.

7. MULTIPLE DIMENSIONS

Thus far we have only looked at line segments in a plane. This decision is reasonable, since the two-dimensional case is by far the most important in practice. However, the techniques we have developed here can conveniently be extended to multiple dimensions. We briefly discuss how in this section.

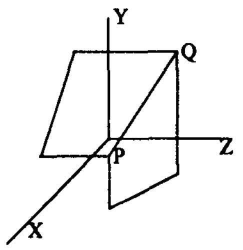

Figure 12. A line segment PQ in 3-space, and its projections to the XY and XZ planes

An infinite line has only one degree of freedom irrespective of the dimension of the space it is in. Given one coordinate of a point on the line, all the other coordinates can be derived as (linear) functions of the given coordinate[4]. For example, in three-dimensional space a line can be represented as a pair of equations of the form $<y = m_y x + b_y,\ z = m_z x + b_z>$. These equations are obtained by projecting the line on the X-Y and X-Z planes respectively, and then writing down the equation of the projected line in the two-dimensional plane. If, in addition to the four parameters in these equations, we also store the X-range of a line segment, then we have a complete definition of a line segment. See Fig. 12.

In k-dimensional space, a line segment is traditionally represented by a $2k$-tuple comprising the coordinates of its two end-points. Instead we have a $2k$-tuple $<m_1, b_1, m_2, b_2, \ldots, m_{k-1}, b_{k-1}, x_{k_{min}}, x_{k_{max}}>$, where the last two entries are the range spanned by the line segment on the x_k axis, and the remaining entries are the slopes and intercepts of the projections of the infinite extension of the given line segment on the x_1–x_k plane, x_2–x_k plane, etc.

Some slopes could become infinite, for certain lines, and ranges may not be very meaningful along axes orthogonal to a given line segment. To avoid such problem cases, we partition the line segments, as in Section 4. k partitions are created in k-dimensional space, one corresponding to each axis. Each line segment is placed in a partition with whose axis it subtends the smallest angle. This axis is the distinguished axis against which all the slopes are determined, as well as the segment range. Since a line cannot be orthogonal to all the axes in space, we are guaranteed that it will always be able to find such an axis, on which a line segment subtends a non-zero range, and which can be used to define slopes along other axes. Moreover, since the line is closer to this axis than to any other axis, in the projection of the line segment to a plane defined by this axis and another one, the slope of the line segment will always be less than or equal to one, closer to this axis than to the other one. Thus all slopes are bounded by ± 1, as in the two-dimensional case.

A point in the original space transforms to a line in each transform space, just as in the two-dimensional case. For any given value of slope, in the projection to a pair of axes, one can determine the appropriate intercept for a line that goes through the point. These functions of slope and intercept along $k-1$ different projections, provide $k-1$ orthogonal projections of a line. From these, a line in k-dimensional space can be determined uniquely.

Finding all line segments that go through a point once again involves finding all line segments that transform to a point on the line (in one of the k transform spaces) obtained by transforming the point, and that also satisfy a range condition.

Finding all line segments that intersect a given query line segment once again involves taking the transforms of the two end points of the query segment. These two lines form a plane, and a region in this plane can be determined as lying *between* these lines, as before. The desired line segments all transform to points in this region.

8. CONCLUSIONS

In this paper we have proposed a data structure to store line segments in a multi-dimensional space. The data structure stores the slope, intercept, and range along one axis of a line segment, rather than storing the end-points. The size of the storage structure is roughly the same as for conventional techniques.

We have shown how this proposed storage structure can be used effectively to retrieve the set of line segments in a database that (i) pass through a point, (ii) pass close to a point, and (iii) intersect a given line segment. We have also demonstrated, through geometric analysis and simulations, that our method can provide a significant improvement over traditional techniques that store end-points or use bounding rectangles.

4. Barring degenerate cases. We shortly show how degenerate cases can be avoided.

ACKNOWLEDGEMENT

This problem was suggested to me by Larry O'Gorman, who also provided the references to the image recognition literature. This problem was also suggested to me independently by Kyu-Young Whang. I had useful discussions with Shaul Dar and Larry O'Gorman.

APPENDIX

Intersection Calculations for a Line Segment Query

Consider a query segment with end points $P=(x_1,y_1)$ and $Q=(x_2,y_2)$. For every value of $m \in [-1,1]$, we can compute a range for the allowable intercepts from the equations: $b_1 = -x_1 m + y_1$ and $b_2 = -x_2 m + y_2$. Any line that intersects this query segment has a b value within this range of limits for every choice of m. Observe that when $m = m_0$, the slope of the query segment, then $b_1 = b_2 = b_0$, the intercept of the query segment, and the range becomes a unique point.

A line with slope m and intercept b_1 intersects the line at point P. A line with slope m and intercept b_2 intersects the line at point Q. A line with slope m and an intercept b, between these two limits, intersects the line at a point with X-coordinate determined by a simple ratio: $x_{intersection} = x_1 + \frac{x_2 - x_1}{b_2 - b_1}(b - b_1) = x_1 + \frac{b - b_1}{m_0 - m} = \frac{b - b_0}{m_0 - m}$. This coordinate, $x_{intersection}$, must lie within the bounds of the data line segment under consideration for it to be retrieved using the exact method. If the query line segment is vertical, m_0 is meaningless, but $x_1 = x_2 = x_{intersection}$.

For data line segments in $\mathbf{D_Y}$, similarly, one can compute bounds on the c values as a function of a specified n value. And an exact $y_{intersection}$ as a function of the c value and n value.

Intersection Calculations for a Region Query

Consider a query rotated square, centered at a point (x_0,y_0), with a side of length $\sqrt{2}k$. The top vertex of this square has coordinates $(x_0, y_0 + k)$, and the bottom has coordinates $(x_0, y_0 - k)$. Considering line segments in $\mathbf{D_X}$, we find that lines intersecting the "vertical" diagonal between these top and bottom points must have an intercept b in the range $[y_0 - mx_0 - k,\ y_0 - mx_0 + k]$ if the slope is m.

The exact range condition requires that we consider cases. A line segment in $\mathbf{D_X}$ intersects the given rotated square region only if its right end point is to the right of the left segments of the rotated square, and its left end point is to the left of the right segments. In other words, x_{max} should be greater than or equal to x_{li}, and x_{min} less than or equal to x_{ri}, where x_{max} and x_{min} are the X coordinates of the end-points of the candidate line segments, x_{li} is the X coordinate of the intersection of (the infinite extension of) the line segment with one of the two left side edges of the rotated square as appropriate, and x_{ri}, is similarly the X coordinate of the right intersection.

The lower left edge of the rotated square has an equation: $y = x_0 + y_0 - k - x$. If this line segment intersects with the candidate line segment: $y = mx + b$, the intersection coordinates are $(\frac{x_0 + y_0 - k - b}{m+1}, \frac{m^*(x_0 + y_0 - k) + b}{m+1})$. For this intersection to occur within the lower left edge, we must have: $x_0 - k \le x_{li} \le x_0$ and $y_0 - k \le y_{li} \le y_0$. Due to the linear relationship between x and y coordinates on this line segment, we need check this condition only for one of x or y. Using the constraint on y, we derive the following limits on the intercept of the candidate line segment: $y_0 - mx_0 - k \le b \le y_0 - mx_0 + mk$. The first inequality is automatically satisfied due to the limits on intercept value already placed. We thus say, that when the second inequality is satisfied, we must have $x_{max} \ge \frac{y_0 - b + x_0 - k}{m+1}$. Performing similar calculations over each of the four edges of the rotated square, we get the following range conditions:

$$x_{max} \ge \frac{y_0 - b + (x_0 - k)}{m+1} \quad \text{if } b \le y_0 - m(x_0 - k)$$
$$x_{max} \ge \frac{y_0 - b - (x_0 - k)}{m-1} \quad \text{if } b \ge y_0 - m(x_0 - k)$$
$$x_{min} \le \frac{y_0 - b - (x_0 + k)}{m-1} \quad \text{if } b \le y_0 - m(x_0 + k)$$
$$x_{min} \le \frac{y_0 - b + (x_0 + k)}{m+1} \quad \text{if } b \ge y_0 - m(x_0 + k)$$

A similar set of conditions can be obtained for line segments in $\mathbf{D_Y}$ by interchanging x and y, and substituting n for m and c for b everywhere.

REFERENCES

[1] I. J. Cox, J. B. Kruskal, and D. A. Wallach, "Predicting and Estimating the Performance of a Sub-pixel Registration Algorithm," *IEEE Trans. on Patern Analysis and Machine Intelligence*, to appear, 1990.

[2] A. Guttman, "R Trees: A Dynamic Index Structure for Spatial Searching," *Proc. ACM SIGMOD Int'l Conf. on the Management of Data*, 1984, 47-57.

[3] A. Henrich, H-W. Six, and P. Widmayer, "The LSD Tree: Spatial Access to Multidimensional Point and Non-point Objects," *Proc. 15th Int'l Conf on Very Large Databases*, Amsterdam, The Netherlands, 1989, 45-54.

[4] K. Hinrichs and J. Nievergelt, "The Grid File: A Data Structure to Support Proximity Queries on Spatial Objects," Tech. Report 54, Institut fur Informatik, ETH, Zurich, July 1983.

[5] P. V. C. Hough, "Method and Means for Recognizing Complex Patterns," U. S. Patent No. 3069654, 1962.

[6] H. V. Jagadish, "Spatial Search with Polyhedra," *Proc. Sixth IEEE Int'l Conf. on Data Engineering*, Los Angeles, CA, Feb 1990.

[7] R. Kasturi and J. Alemany, "Information Extraction from Images of Paper-Based Maps," *IEEE Trans. on Software Engineering*, SE-14(5), May 1988.

[8] R. C. Nelson and H. Samet, "A Consistent Hierarchical Representation of Vector Data," *Computer Graphics*, 20(4), Aug. 1986, 197-206.

[9] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The Grid file: An Adaptable Symmetric Multikey File Structure," *ACM Trans. on Database Systems*, 9(1), 1984.

[10] L. O'Gorman and G. I. Weil, " "An Approach Towards Segmenting Contour Line Regions"," *Proc. 8th Int. Conf. Pattern Recognition*, Paris, France, 1986, 254-258.

[11] J. A. Orenstein and T. H. Merett, "A Class of Data Structures for Associative Searching," *Proc. Third SIGACT News SIGMOD Symposium on the Principles of Database Systems*, 1984, 181-190.

[12] J. A. Orenstein, "Redundancy in Spatial Databases," *Proc. ACM SIGMOD Int'l Conf. on the Management of Data*, Portland, OR, May-June 1989.

[13] N. Roussopoulos, C. Faloutsos, and T. Sellis, "An Efficient Pictorial Database System for PSQL," *IEEE Trans. Software Engg.*, 14(5), May 1988, 611-629.

[14] H. Samet and R. E. Webber, "Storing a Collection of Polygons Using Quadtrees," *ACM Trans. on Graphics*, 4(3), July 1985, 182-222.

[15] H. Samet, "Hierarchical Representation of Collections of Small Rectangles," *ACM Computing Surveys*, 20(4), December 1988, 271-309.

[16] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+ Tree: A Dynamic Index for Multidimensional Objects," *Proc. 13th Int'l Conf on Very Large Databases*, Brighton, U. K., Sep. 1987, 507-518.

[17] M. Tamminen, "Performance Analysis of Cell-based Geometric File Organizations," *Computer Vision, Graphics, and Image Processing*, 24, 1983, 160-181.

[18] P. M. Will and K. S. Pennington, "Grid Coding: A Preprocessing Technique for Robot and Machine Vision," *Artificial Intelligence*, 2(3-4), 319-329.