NISTIR 5447

A Certainty Grid to Object Boundary Algorithm

John Albert Horst
Hui-Min Huang
Tsung-Ming Tsai
Intelligent Systems Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

NIST

NISTIR 5447

A Certainty Grid to Object Boundary Algorithm

John Albert Horst
Hui-Min Huang
Tsung-Ming Tsai
Intelligent Systems Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

June 1994

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arati Prabhakar, Director

A CERTAINTY GRID TO OBJECT BOUNDARY ALGORITHM

By John Albert Horst[1], Hui-Min Huang[2], and Tsung-Ming Tsai[3]

ABSTRACT

An edge linking algorithm [Nevatia 80] is modified for a mobile robot map representation application. Certainty grid 'images' are transformed to an appropriate set of object boundary curves. The latter are expressed as oriented piecewise linear segments. Image processing techniques, such as edge detection, thinning, curve tracing, and linear approximation are employed with various modifications. The most significant modification is a new method for linear curve approximation that is simple, accurate, and efficient. This method monitors chord and arc length and its excellent performance is demonstrated against similar algorithms. The certainty grid to object boundary algorithm is tested against simulated noisy certainty grid maps.

1. INTRODUCTION AND MOTIVATION

Various robotic tasks require a variety of map representations. Intelligent systems are well served by as many representations as possible without suffering unacceptable cost or degradation in performance. With processor and memory costs continuing to plummet and with the increasing availability of parallel bus architectures, maintaining multiple representations in real-time intelligent systems is becoming more and more practically feasible. Two useful and complementary map representations are certainty grids [Moravec 88] and object boundary curves. They are particularly useful for mobile robot control. For example, vectors normal to an object boundary would be difficult to get from a certainty grid, but relatively easy to obtain from an object boundary curve. Spatial occupancy information is gotten easily from certainty grids but not so easily from object boundary curves. Additionally, representing spatial occupancy in the form of object boundary curves is important for the detection of higher level features such as corners, curves, and lines. As a result, high level geometric features can be more easily computed, perceived, and updated and, for example, can be used by the mobile robot to reorient itself. This is particularly important when hierarchical control architectures are used [Albus 91]. It is relatively easy to build and maintain a certainty grid which makes it a good local map. However, a certainty grid representation for a global map may require a forbidding amount of storage space, whereas, an object boundary curve is more compact without sacrificing accuracy or utility.

We describe an algorithm that receives a certainty grid and outputs a set of oriented contiguous line segments that approximate the object boundary curves in the certainty grid. The certainty grid to object boundary (CGOB) algorithm we describe uses edge operators obtained from [Jain 89] where they are demonstrated to have good performance with noisy images. This reduces the importance of the noise thresholding parameter in Nevatia and Babu's algorithm [Nevatia 80].

The CGOB algorithm concludes with a piecewise linear curve approximation algorithm. The popular split and merge approach [Chen 79, Duda 73, Jain 89, Grimson 90] is known to be inefficient and several attempts to improve its efficiency

[1]Electronics engineer.

[2]Mechanical engineer.

[3]Mechanical engineer.

Robot Systems Division, The National Institute of Standards and Technology (NIST), U.S. Department of Commerce

come with an increase in complexity [Nevatia 80, So 93]. We have developed a new approach that is accurate, simple, and efficient. We compare this new approach to others in the literature. We compare the chord and arc length algorithm to other approaches in the literature and its efficient and accurate performance argues for its inclusion in the CGOB algorithm we describe.

The CGOB algorithm has been tested against certainty grids of various types corrupted by two types of noise (see figure 9). The points that represent the object boundary are also ordered so that occupied area is always to the right if proceeding in the direction of this list of points.

Hierarchical control (such as RCS [Albus 91]) specifies a real-time, multi-level interaction of prediction and error formation. The certainty grid representation allows prediction of points (low level), whereas the object boundary representation allows prediction of lines and shapes (higher level). The current algorithm will allow both representations to simultaneously exist and be updated in a real-time intelligent system.

2. THE ALGORITHM

Here is the CGOB algorithm (based on Nevatia & Babu):

1) Create a raw edge grid using two orthogonal 5x5 gradient operators on a noisy certainty grid.
2) Threshold and thin the raw edge grid and use this result to compute arrays of 'predecessors' and 'successors'.
3) Group contiguous cells in the thinned edge grid, constituting contiguous edge cells.
4) Do local Gaussian smoothing on each group of points (to filter out quantization noise).
5) Approximate the smoothed boundary points with contiguous line segments by monitoring change in chord length and path length.

A two dimensional view of the world is sufficient for many navigation tasks of a mobile robot. However, we suggest that the algorithm can be extended into three dimensions. This will be discussed further in the conclusion.

When using connected line segments (as the object boundaries) to represent spatial occupancy, we are particularly interested in specifying the ordering of each object boundary curve so that it encodes which side of the curve is occupied.

Certain aspects of the algorithm are simplified and standardized because we are always in unitary (or 'grid') coordinates. Distances will always be 1 or $\sqrt{2}$ in grid units.

2.1 EDGE DETECTION

Nevatia & Babu used six 5x5 masks as 'compass' operators [Jain 89] (*i.e.*, choose the signed gradient orientation associated with the maximum value as the gradient). Therefore, the output orientation at each cell is one of twelve discrete values. We used two orthogonal masks for computing the gradient. This gave a smoother gradient function with less computational cost.

We used 5x5 stochastic gradient operators [Jain 89]. These operators have the advantage of performance tailored to the expected signal to noise characteristics of the raw certainty grid. This signal to noise ratio (SNR) needs to be computed from a representative noisy certainty grid in order to be accurate. We performed our simulations with a somewhat low SNR of one.

3x3 operators didn't produce smooth thinned edges, so 5x5 operators were required. In addition to requiring more computation (about three times more), 5x5 gradient operators caused the loss of fine detail. However, for our purposes, assurance of smooth thinned edges is more important than either higher computational cost or loss of detail. The presence of 'forks' in the edge grid are an example of such detail. Even though the Nevatia & Babu algorithm is capable of handling forks, we were unable to create a simulated certainty grid that produced anything other than spurious forks when using 5x5 gradient operators.

2.2 THRESHOLDING AND THINNING

After edge detection, a thresholding operation is used to eliminate spurious edges. Our choice of the stochastic gradient operator made the choice of this threshold value less critical, since the stochastic gradient eliminates much of the noise.

The thinning step seeks to find the cells associated with the true edge and eliminate all others. The orientation and magnitude of each cell is examined. We classify the edge orientation as lying within one of four like-shaded regions in figure 1. The non-maximum suppression algorithm is employed for thinning and requires the following for a cell to be an edge:

1) the cell magnitude is a local maxima with respect to its two neighbors orthogonal to the edge orientation (see figure 1).
2) the difference in edge orientation of this cell with its two neighbors is less than threshold ($\pi/3$ worked well with our simulations)

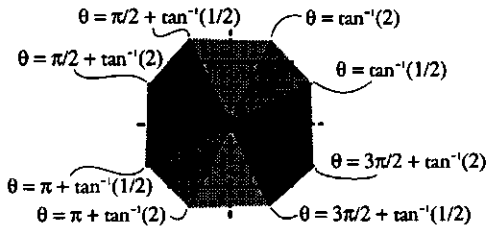

Figure 1: Regions of edge orientation in the neighborhood of a cell.

If the cell passes both the threshold and the thinning test, it is stored in the thinned edge grid and the two neighbors are excluded from consideration as potential edge cells.

2.3 CURVE LINKING

Now that the thinned edge grid is formed, we must order these cells into sets of ordered lists which describe the curves (closed, open, and forks) that we expect to see. We followed Nevatia & Babu by forming successor and predecessor grids that contain the chosen successor and predecessor for each cell. These predecessors and successors are then used to get linked lists of curves representing the object boundaries.

2.3.1 PREDECESSORS AND SUCCESSORS

The predecessors and successors for each edge cell in the thinned edge grid are chosen as follows.

1) Determine within which of the eight regions (as in figure 1) the edge orientation lies (edge direction is defined $\pi/2$ counterclockwise from the edge gradient direction).
2) With this information, define potential successors and predecessors (three each maximum) based on whether those potential cells are in the thinned edge grid.
3) If there are predecessors, record that fact. Any more information on predecessors is not necessary.
4) If there is only one successor (defined in the edge direction), just choose it.
5) If there are exactly two successors, and
   a) if the potential successor cells are not 4-neighbors, choose one with the greatest magnitude as the successor and store other as a fork.
   b) if the potential successor cells are not 4-neighbors, choose one as a potential fork only if its edge orientation differs by more than threshold (we used $\pi/3$). Otherwise, choose the nearest of the two as successor (Euclidean distance).
5) If there are exactly three successors,
   a) the one with maximum difference in edge orientation is chosen as a potential fork.
   b) successor is the closest one by Euclidean distance.

Optimum paths are not sought after as in dynamic programming techniques [Bellman 62, Jain 89], because of the increase of computation required.

The edge direction is defined as $\pi/2$ counterclockwise from the edge gradient direction and the edge direction defines the successor direction. Therefore, occupied space is on the right if one follows successors.

Using a 5x5 edge approximator, features that size or smaller (depending on noise levels) will be missed or misinterpreted. This is probably why we were never able to create forks in our simulations. Nonetheless, we retained the code to detect forks from Nevatia & Babu.

2.3.2 CURVE GENERATION

Now that we have grids of successors, predecessors, and forks, we need to exploit these grids to obtain the ordered lists that constitute the object boundary curves in the

certainty grid. Curve generation requires two serial passes through the grid.

1) Look for cells with a successor and no predecessor and store them starter of open curves. At the same time, look for fork cells.
2) Start tracing at each starter cell (excluding fork cells). Eliminate the starter cell from further consideration. If the current cell has a successor and this potential successor cell has not been eliminated, store the successor into the ordered list. Delete each traced edge cell in the thinned edge map.
3) Start tracing at each fork only if the successor of the fork point successor has not been eliminated. Eliminate cells in fork curves.
4) Trace closed curves starting at the first cell encountered that is 'alive'. Eliminate cells from thinned edge grid as they are traced. Continue cycling through the entire thinned edge map.

2.4 PIECEWISE LINEAR CURVE APPROXIMATION

The map is now represented as sets of contiguous grid cells. Each set defines an object boundary curve. This is already a significant reduction in data storage from the certainty grid. However, further reductions can usually be made by approximating these curves with even fewer points. These points then represent the approximation to the object boundary.

One can, of course, use more sophisticated methods of fitting a curve to the set of obstacle boundary points such as splines and higher order polynomials, but line segments have the advantage of simplicity. We avoided optimal linear approximations as in [Pavlidis 77] since they are computationally much more expensive [Ramer 72]. The trade-off of optimality for simplicity and speed we felt was reasonable.

2.4.1 MOTIVATION

Each object boundary is a list of contiguous cells constituting a curve in two dimensions. We wish to obtain a set of points that approximate that curve according to some criterion of fitness. One algorithm that does this is called 'split and merge' [Chen 79, Duda 73, Jain 89, Grimson 90]. It uses a minimum mean squared error criterion. However, it is known to be inefficient [Nevatia 80, So 93]. This is because the size of the curve and the tightness of fit required greatly effect the computational cost. Several attempts have been made to make it more efficient but at the cost of higher complexity [Nevatia 80, So 93]. The split and merge method has some essential weaknesses,

1) It is inefficient because it may require forming approximation errors from the same points on the curve up to $(n-2)(n-1)/2$ times for an n point curve (during the 'split' portion of the algorithm).
2) It tends to inscribe curves, requiring more approximating points for a tighter fit.

The strengths of the split and merge algorithm are its simplicity, that it is controlled by a single, physically meaningful parameter, and that it generates a sparse set of points that well approximate the original curve (as in figure 5).

Teh and Chin [Teh 89] have also created an algorithm for finding dominant points on digital curves that has a reasonably good error performance. However, their method suffers from the following weaknesses,

1) It is complex to describe and we found the concept of 'region of support' to be somewhat non-intuitive.
2) No parameters are supplied to control tightness of fit.
3) We were able to achieve a better error performance with the chord and arc length algorithm (see figure 7).
4) The algorithm can be performed in parallel, but is relatively slow if done serially.
5) It assumes closed, digital curves only.

We sought an expression that can be computed in a single pass through the curve but still minimizes mean squared error. Such requirements are satisfied by monitoring the relationship of chord length and arc length along the curve.

2.4.2 CHORD AND ARC LENGTH ALGORITHM

The chord and arc length algorithm we now describe avoids all the stated weaknesses of both the split and merge and the Teh-Chin algorithms. It has the following processing steps,

1) Determine whether the curve is open or closed.
2) Do local Gaussian smoothing on the raw data to reduce quantization error.
3) Starting anywhere on the closed curve (at the first point on the open curve), compute chord length, C, and arc length, S, for each successive point and if $1/2\sqrt{S^2-C^2}$ is greater than the maximum deviation parameter, declare the previous point to be dominant.

4) Merge points by testing if approximating points can be eliminated without exceeding the threshold on deviation.
5) Compute a (parameterized) least squares line to the points on the curve between and including the last two dominant points.
6) Find the point on the previous and current least squares fit lines that are closest to the previous dominant point.
7) Choose the midpoint between these two closest points as the latest approximating point.

Gaussian smoothing is required only because the thinning algorithm often produces a thinned edge pixel having one four-neighbor and one eight-neighbor. Gaussian windows of size five[4] were used in our simulations. Closed curves do operations mod(n) for n points on the curve. The start and end points of open curves are automatically declared to be dominant points. We are free to eliminate least squares line computation (steps 4-6), in which case, the chord and arc length algorithm is very Spartan indeed, as well as efficient, but would not ameliorate the 'inscribing' problem.

The algorithm has the following characteristics,

1) It is efficient because the number of computations increase only linearly with the number of points in the curve.
2) It uses the same physically meaningful parameter used by the split and merge algorithm, namely, maximum deviation.
3) It has excellent error performance against several algorithms, namely, low error per number of approximating points (as in figure 7).
4) It provides a better fit to smoothly curving data (e.g., circles) because it computes least squares lines and uses them to compute the approximating points.
5) The only additional complexities over the split and merge method are the addition of the least squares line computation and Gaussian smoothing, neither of which are required. The core part of the chord and arc length approach is even simpler to describe and code than the split and merge.
6) It is designed for both open and closed curves.
7) It seems to be insensitive to the particular choice of starting point.

[4]Gaussian windows of size three, five, and seven were {0.1586, 0.6827, 0.1586}, {0.0228, 0.22978, 0.4950, 0.2297, 0.0228}, {0.0062, 0.0606, 0.2417, 0.3829, 0.2417, 0.0606, 0.0062}.

2.4.3 ANALYSIS

We now describe the chord and arc length algorithm more precisely. Let $\alpha(i)$, $i=1,2,\ldots,n$, be a sequence of points in the plane (if $\alpha(1)\approx\alpha(n)$ we say the curve is closed). We seek a subsequence of dominant points, $\alpha(i_j)$, $j=1,2,\ldots,m\le n$ such that, for $j=1,2,\ldots m$ and $i_j<i<i_{j+1}$,

$$d\ge\max\left\{\text{distance}\left(\alpha(i),\text{line segment}\left(\alpha(i_j),\alpha(i_{j+1})\right)\right)\right\}, \quad (1)$$

for a given maximum deviation value, d. The distance function computes the shortest distance from the points $\alpha(i)$, $i_j<i<i_{j+1}$, to points on the line segment between adjacent dominant points $\alpha(i_j)$ and $\alpha(i_{j+1})$. With i_j as the previously determined dominant point, we form expressions for chord length and arc length between the points, $\alpha(i_j)$ and k,

$$c(i_j,k)=\left\|\alpha(k)-\alpha(i_j)\right\|, \quad (2)$$

and

$$s(i_j,k)=\sum_{i=i_j}^{k-1}\left\|\alpha(i+1)-\alpha(i)\right\|, \quad (3)$$

respectively, until

$$d(i_j,k)=\frac{1}{2}\sqrt{s^2(i_j,k)-c^2(i_j,k)}>d. \quad (4)$$

Then we choose $i_{j+1}=k-1$ (the previous point) as the new dominant point. Therefore, since the function, $d(\cdot)$, is monotonically increasing,

$$d(i_j,i_{j+1})\le d. \quad (5)$$

If the curve is closed, all operations are mod n; if open, choose $\alpha(1)$ and $\alpha(n)$ as the first and last dominant points.

We must show that (1) is true if condition (4) is met throughout the sequence of points $\alpha(i)$. This can most easily be demonstrated by examination of figure 2. The equation,

$$d(i_j,i_{j+1})=\frac{1}{2}\sqrt{s^2(i_j,i_{j+1})-c^2(i_j,i_{j+1})}, \quad (6)$$

comes from noting that, if the curve were a string attached at the chosen dominant points $\alpha(i_j)$ and $\alpha(i_{j+1})$ and we pushed the string upwards with a stick, we could always form an isosceles triangle such as that in figure 2.

A illustrated in figure 2, it is true that for any planar curve,

$$e_{\max}(i_j, i_{j+1}) \le d(i_j, i_{j+1}), \qquad (7)$$

where $e_{\max}(i_j)$ is the largest deviation from points $\alpha(i)$, $i_j < i < i_{j+1}$, to the line segment between adjacent dominant points $\alpha(i_j)$ and $\alpha(i_{j+1})$. We prove (7) in the appendix.

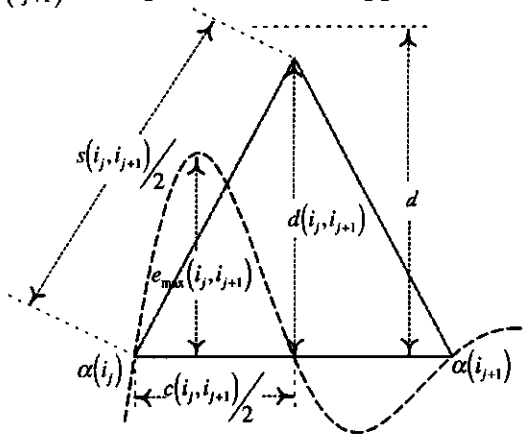

Figure 2: The isosceles triangle is formed from the chord length and arc length between the two points $\alpha(i_j)$ and $\alpha(i_{j+1})$ on the curve.

Since (7) is true for all $j = 1,2,\ldots m$ and from (5),

$$e_{\max}(i_j, i_{j+1}) \le d, \qquad (8)$$

for all $j = 1,2,\ldots m$, and (1) must also be true.

A lemma and a theorem (stated and proved in the appendix) reveal that if we are given a threshold value, d, the basic chord and arc length algorithm we have described will guarantee that the deviation of each point on the curve from its appropriate approximating line segment (i.e., the chord line segment) will be less than or equal to d. This provides a physically meaningful upper bound on the approximation while maintaining efficiency.

The merge process operates as in the split and merge method [Grimson 90], namely, if

$$d \ge \max\left\{\text{distance}\left(\alpha(i), \text{line segment}\left(\alpha(i_{j-1}), \alpha(i_{j+1})\right)\right)\right\}, \qquad (9)$$

for $i_{j-1} < i < i_{j+1}$, $\alpha(i_j)$ is eliminated as a dominant point. What we have described thus far is the basic chord and arc length approach. Next we describe the extended algorithm including Gaussian smoothing and least squares fitting.

Gaussian smoothing is not a necessary part of this approach, but can be useful with digital curves when we wish to eliminate quantization noise [Horst 94]. Let $\beta(i)$, $i = 1,2,\ldots,n$, be the unsmoothed sequence of points in the plane. We let

$$\alpha(i) = \sum_{j=i-\frac{w-1}{2}}^{i+\frac{w-1}{2}} \beta(j) G(j-i), \qquad (10)$$

be the smoothed values for Gaussian window size, w, an odd number. The values $G(\cdot)$ are integrations of appropriately chosen, normalized one-dimensional Gaussian functions.

We now describe a least squares fitting technique to further reduce approximation error. Like Gaussian smoothing, least squares fitting is not a necessary part of this approach, but may be useful if computing time is available and analog points are acceptable. Each subset of points $\{\alpha(i_j), \alpha(i_j+1), \ldots, \alpha(i_{j+1})\}$ is used to fit a parameterized line[5], $l_2(t)$,. We find points, p_1 and p_2, on the previous least squares line, $l_1(t)$,, and the current line, $l_2(t)$,, which are the shortest distance from $\alpha(i_j)$. The latest approximating point is chosen as the midpoint between p_1 and p_2[6].

We note that the upper bound used in the chord and arc length method is not as tight an upper bound as in the split and merge approach, since it is an indirect measure of error. However, the efficiency and excellent error performance of the chord and arc length approach argue for its utility.

2.4.4 ALGORITHM PERFORMANCE

It is essential to examine the error between the approximating line segments and the smoothed curve points as a function of the number of approximating points for a variety of types of curves. We find that the chord and arc length algorithm performs better than both the split and merge and the

[5]Parameterized fitting easily allows fitting to arbitrary curves and minimizes perpendicular distance.

[6]Some minor changes to this general process are required at the beginning and end of each curve and whether they are open or closed.

Teh-Chin algorithm. We demonstrate the errors of the different algorithms for the digital closed curve of a chromosome as shown in figures 3 through 6. Figure 7 shows maximum actual deviation for the various algorithms for the digital curve as a function of number of approximating points. Results on total deviation and standard deviation errors give similar results. We expect that the split and merge method would do a little better when the starting point of a closed curve is chosen carefully [So 93], however, as we mentioned above, this comes at a cost of greater complexity. Figure 8 illustrates the superior efficiency of the chord and arc length approach over the split and merge algorithm and its insensitivity to the number of approximating points (i.e., 'tightness' of fit).

Note that the maximum deviation threshold used in the split and merge algorithm fulfills the same role in the chord and arc length algorithm we describe. Threshold values will be chosen based on the application (with its unique sensors and geometric shapes and required precision).

Operating times for the chord and arc length algorithm are linearly proportional to the number of data points processed. Because split and merge operating times are proportional to the square of the number of data points, curves with many points will take forbiddingly long times to compute. Additionally, compute time of the split and merge method varies with the size of the deviation threshold (as in figure 8). Alternatively, performance times of the chord and arc length algorithm are constant with respect to the deviation threshold. Another advantage of the chord and arc length algorithm is that it seems to be insensitive to the choice of the initial breakpoint, unlike split and merge [So 93]. A disadvantage of the chord and arc length algorithm is that the difference between the maximum error (resulting from the approximation) and the input deviation threshold is greater than with split and merge because the former computes error indirectly.

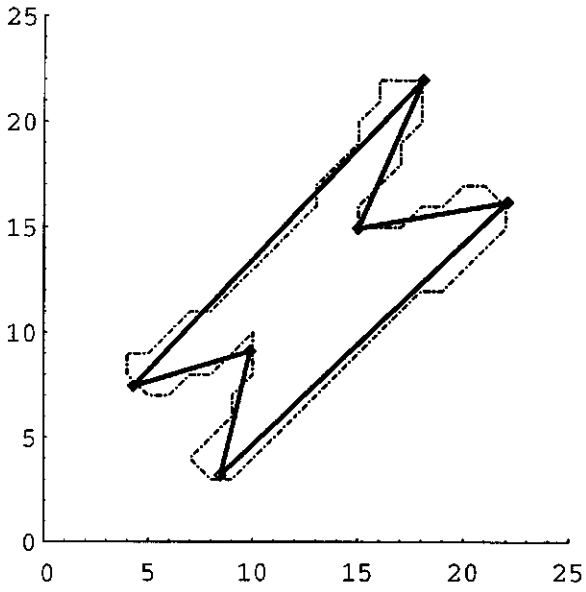

Figure 3: A digital closed curve of a chromosome with fitting by chord and arc length with least squares fitting.

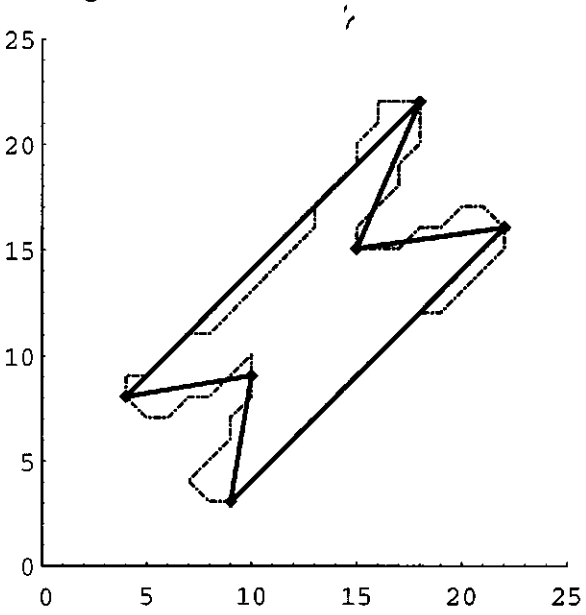

Figure 4: A digital closed curve with fitting by chord and arc length without least squares fitting.

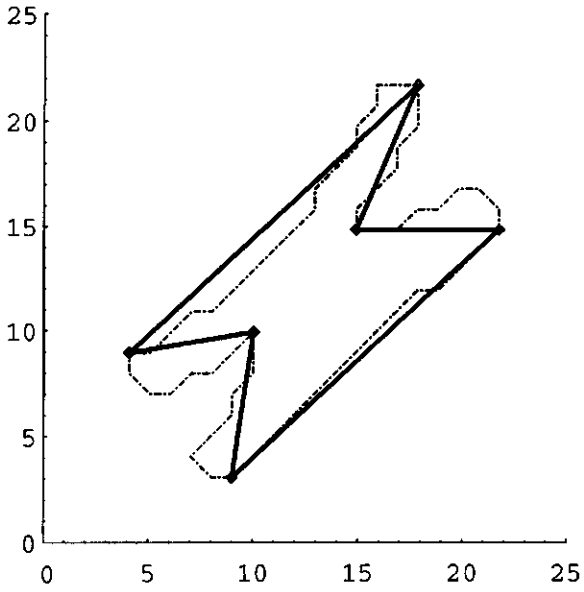

Figure 5: A digital closed curve with fitting by split and merge.

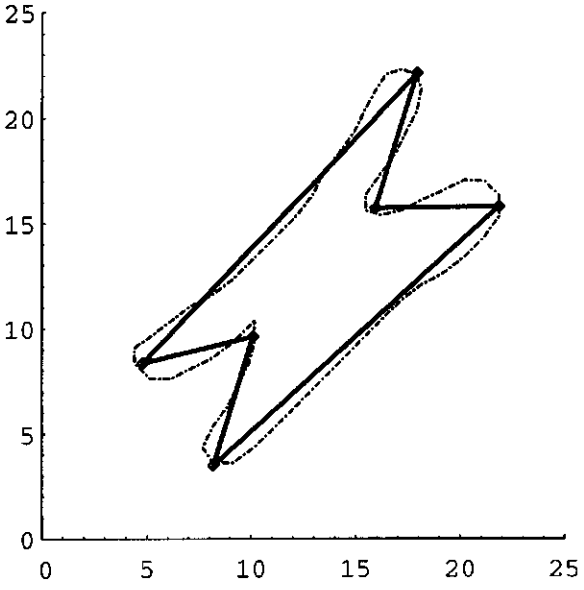

Figure 6: A Gaussian smoothed (window size = 5) closed curve with fitting by chord and arc length with least squares fitting.

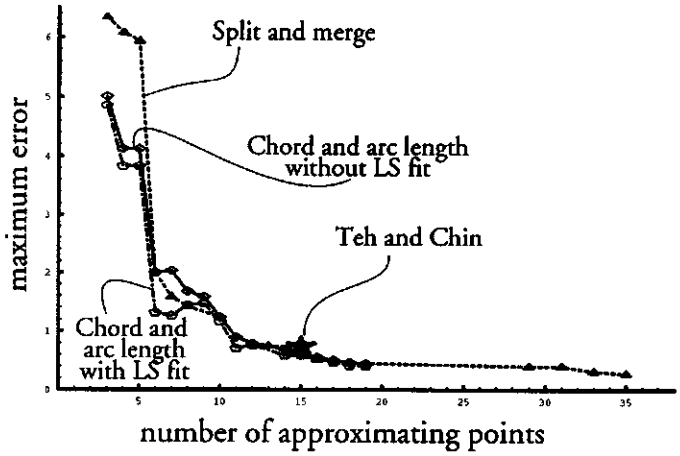

Figure 7: Maximum errors as a function of the number of approximating points for the digital curve in figure 3.

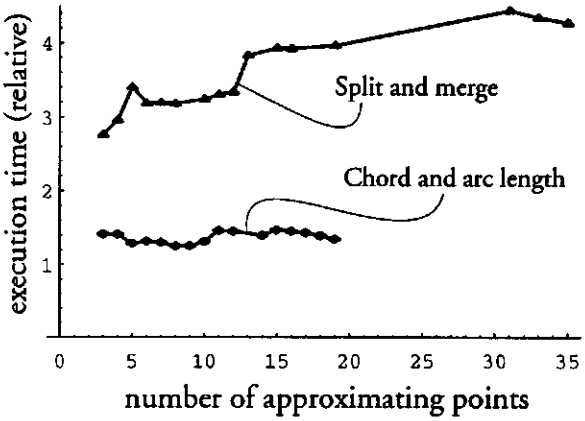

Figure 8: Relative execution time as a function of the number of approximating points for the chord and arc length algorithm and the split and merge method using the curve in figure 3.

2.5 SIMULATION

For testing the algorithm, we used simulated grid maps with Gaussian blurring and speckle noise added. Figure 9 shows an example of a noisy grid. Note that the final approximating line segments are superimposed. Here we see that open or closed curves can be detected and that data reduction from the certainty grid to the obstacle boundary points is about 145 in this example. Figures 10 and 11 show the raw edge grid and the thinned edge grid. The entire CGOB algorithm has been tested and exists in both Mathematica® and 'C'.

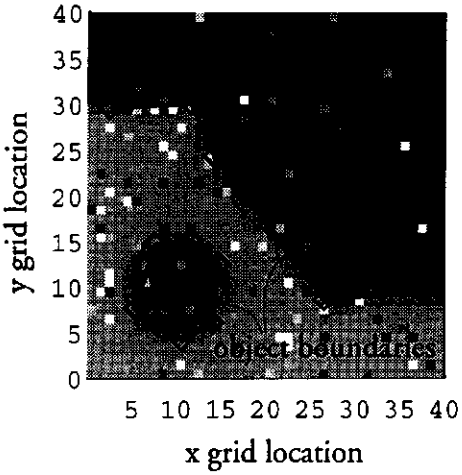

Figure 9: A noisy certainty grid with final approximating points superimposed.

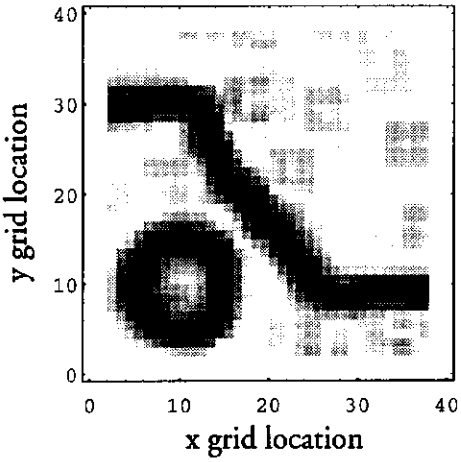

Figure 10: The raw edge grid for noisy certainty grid of figure 9.

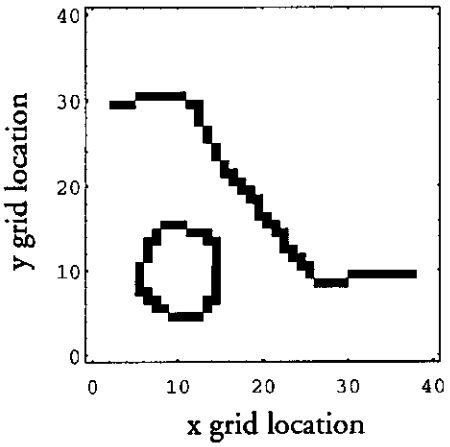

Figure 11: The thinned edge grid for noisy edge grid of figure 10.

3. CONCLUSION

We have made application to mapping (with appropriate changes) of an image processing algorithm by Nevatia and Babu that converts an intensity image into sets of connected line segments representing the edges in the image. It is robust in the presence of blurring and 'speckle' noise as long as the gradient operator is chosen carefully. Its use in a real-time system can enhance that system's efficient performance of a variety of tasks. In summary, here are some of the uses and advantages of having the CGOB algorithm in a real-time system:

1) It allows simultaneous use and dynamic availability of two forms of spatial occupancy representation, namely, certainty grids and object boundary curves.
2) Certainty grids provide a fast response to some queries that would take longer to get with object boundary curves (e.g., Is there an object at (x,y)?). The latter would have faster response to other types of queries (What is the normal vector to the boundary at (x,y)?).
3) Certainty grids require much more storage space than object boundary curves. This fact might suggest the former for local, egocentric maps and the latter for global maps.
4) Object boundary curves can be considered a 'higher level' representation in an intelligent hierarchical system, since, for example, it is easy to detect geometric shapes using object boundary curves.

The CGOB algorithm is described for two dimensional maps only, but the move to three dimensions should be straightforward and is a topic for further research. Particularly so, the extension of the chord and arc length algorithm to three dimensions is worthy of further pursuit. Edge detection and linking could logically be extended to three dimensions and approximation by line segments in R^2 would become approximation by flat planes R^3, perhaps described by three points. Edge detection would be done with three block matrices that would approximate partial derivatives of the occupancy function in 3-space in the x, y, and z directions. Linking in 3D would not be line linking but surface linking. Some clever modification of the successor and predecessor file idea would be needed.

If CGOB continues to show promise as we test it within real-time robotic systems, we would probably want to develop an algorithm that works in the reverse direction,

namely, to generate a certainty grid from an object boundary map. For example, we may be confident that at the object level, we have sensed object A (precise model stored in memory) and can use this information now to improve the certainty grid. How hard is it to go in the reverse direction? This may depend on the complexity of the map.

In order to capture lines, one might want to have a separate step using line detection operators to catch line edges that the edge detection operators might have missed, since the latter should miss objects of size less than 5x5.

Finally, we have shown that the chord and arc length algorithm used for linear approximation has superior error performance, simplicity, and efficiency compared to other approaches in the literature.

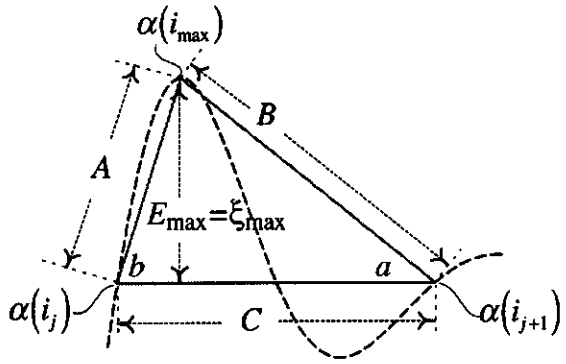

Figure 12: The triangle formed by the point on the curve producing the maximum deviation from the chord line segment.

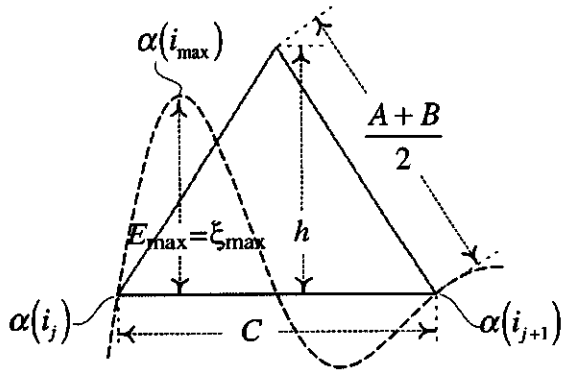

Figure 13: The isoceles triangle 'equivalent' to the triangle of figure 12. The two triangles are equivalent in that the lengths of base and perimeter. are equal

4. APPENDIX

We now prove (7), but first we state some definitions.

Definitions: Let $\alpha(i)$, $i=1,2,\ldots,n$, be a sequence of distinct points in the plane. Let $S = s(i_j,i_{j+1})$, the arc length function (3), $C=c(i_j,i_{j+1})$, the chord length function (2), $E_{max} = e_{max}(i_j,i_{j+1})$, the maximum distance from points $\alpha(i)$, $i=i_j,i_j+1,\ldots,i_{j+1}$ to the line segment formed by the points $\alpha(i_j)$ and $\alpha(i_{j+1})$, let $\alpha(i_{max})$ be the point on the curve where the deviation from the chord line segment is at a maximum (illustrated in figures 12 and 14), ξ_{max} is the distance from $\alpha(i_{max})$ to the chord line formed by the points $\alpha(i_j)$ and $\alpha(i_{j+1})$, and let $D = 1/2(S^2-C^2)^{1/2}$.

Lemma: $\xi_{max} \le D$.

Proof: Construct a triangle formed by the points, $\alpha(i_j)$, $\alpha(i_{max})$, and $\alpha(i_{j+1})$ with base of length C and sides of lengths A and B. ξ_{max} is illustrated in figures 12 and 14. Let b be the angle opposite the side of length B and let a be the angle opposite the side of length A. Note that $\xi_{max} = E_{max}$ only when a and b are less than or equal to $\pi/2$. In figures 12 and 14, ξ_{max} is the height of the triangle. Construct an isoceles triangle, as in figures 13 and 15, with base length and perimeter equal to those of the triangle in figures 12 and 14, respectively. Let h be the height of the isoceles triangle. Let $R = A+B$. Since the shortest distance between two points is a line, $R \le S$, where S is the arc length. From elementary geometry,

$$\xi_{max} = h\sqrt{1-\left(\frac{A-B}{C}\right)^2}, \qquad (9)$$

which implies that $\xi_{max} \le h$. Since $R \le S$ and $h = 1/2(R^2-C^2)^{1/2}$, $h \le D$. Since $\xi_{max} \le h \le D$, $\xi_{max} \le D$. ■

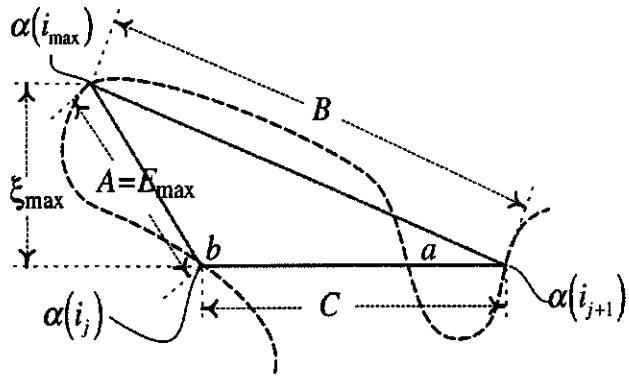

Figure 14: The triangle formed by the point on the curve producing the maximum deviation from the chord line segment where angle b is greater than $\pi/2$.

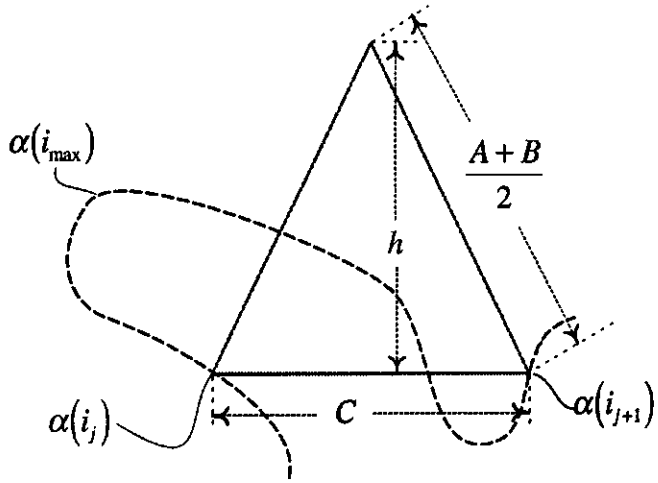

Figure 15: The isoceles triangle 'equivalent' to the triangle of figure 14. The two triangles are equivalent in that the lengths of base and perimeter. are equal

The lemma shows that the distance from $\alpha(i_{max})$ to the chord line is always less than $D = 1/2(S^2 - C^2)^{1/2}$ (the monotonically increasing function of chord and arc length that is the basis of the algorithm). However, if $a \geq \pi/2$, $E_{max} = B$ and if $b \geq \pi/2$, $E_{max} = A$; and in both these cases, $E_{max} \geq \xi_{max}$ (see figure 14). Therefore, the lemma alone is not sufficient to support the claim of (7). In other words, we need to prove that the distance from $\alpha(i_{max})$ to the chord line **segment** is always less than D which is stated in the following theorem.

Theorem: $E_{max} \leq D$.

There are only three cases to consider. We will now show that the theorem is established for each case.

Case 1: $b < \pi/2$ and $a < \pi/2$,

Case 2: $b \geq \pi/2$, and

Case 3: $a \geq \pi/2$.

Proof of Case 1: As in figure 12, if $b < \pi/2$ and $a < \pi/2$, since $E_{max} = \xi_{max}$, the lemma implies that $E_{max} \leq D$. ■

Proof of Case 2: If $b \geq \pi/2$, as illustrated in figure 14, $E_{max} = A$. Construct an isoceles triangle as in figure 15. Let h be the height of this isoceles triangle. From the proof of the lemma, $h \leq D$. By the Pythagorean relation,

$$2h^2 = A^2 + B^2 + 2AB - C^2, \qquad (10)$$

and by the law of cosines (from figure 5),

$$0 = A^2 - B^2 - 2AC\cos b + C^2. \qquad (11)$$

Adding (10) and (11), we get that

$$2h^2 = 2A^2 + (2AB - 2AC\cos b). \qquad (12)$$

The term in parentheses in (12) ≥ 0, since $\cos b \leq 0$. Therefore, $A \leq h$ and, since $h \leq D$, $A = E_{max} \leq D$. ■

Proof of Case 3: The exact same process followed in the proof for Case 2 is required to show that if $a \geq \pi/2$, $B = E_{max} \leq D$. ■

5. REFERENCES

[Albus 91] Albus, J. S., "A Theory of Intelligent Machine Systems", Proc. of IEEE Intelligent Robots & Systems 1991 -- Intelligence for Mechanical Systems, November 1991.

[Bellman 62] Bellman , R. E. and Dreyfus, S., *Applied Dynamic Programming*, Princeton, N.J..: Princeton University Press, 1962.

[Chen 79] Chen, P. C., and Pavlidis, T., "Segmentation by texture using a co-occurrence matrix and a split-an-merge algorithm," *Computer Graphics, Image Processing*, **10** (1979): 172-182.

[Duda 73] Duda, R. O. and Hart, P. E., *Pattern Recognition and Scene Analysis*, New York: John Wiley, 1973.

[Freeman 74] Freeman , H., "Computer Processing of Line Drawing Images," *Computer Surveys* **6**, (March 1974): 57-98.

[Gray 94] Gray, A, *Curves and Surfaces*, in progress.

[Grimson 90] Grimson, W. E. F., *Object Recognition by Computer: The Role of Geometric Constraints*, Cambridge, Massachusetts: The MIT Press, 1990.

[Jain 89] Jain , A. K., *Fundamentals of Digital Image Processing*, Englewood Cliffs, N.J.: Prentice Hall, 1989.

[Martelli 72] Martelli, A., "Edge Detection Using Heuristic Search Methods," *Computer Graphics and Image Processing* **1**, (August 1972): 169-182.

[Moravec 88] Moravec, Hans P., "Sensor fusion in certainty grids for mobile robots", AI Magazine, Summer 1988, 61-74.

[Rosenfeld 82] Rosenfeld, A. and Kak, A. C., *Digital Picture Processing*, New York: Academic Press, 1982.

[Nevatia 80] Nevatia, R. and Babu, K. R., "Linear Feature Extraction and Description," *Computer Graphics and Image Processing* **13**, 257-269.

[Pavlidis 77] Pavlidis, Theodosios, "Polygonal approximations by Newton's method", *IEEE Transactions on Computers* **C-26**, No. 8, August 1977.

[Ramer 72] Ramer, Urs, "An iterative procedure for the polygonal approximation of plane curves", *Computer Graphics and Image Processing* **1**, 1972, 244-256.

[So 93] So, W. C. and Lee, C. K., "Invariant line segmentation for object recognition", *Proceedings of IECON '93*, Maui, Hawaii, Nov. 15-19, 1993, 1352-1356.

[Teh 89] Teh, Cho-Huak and Chin, Roland T., "On the detection of dominant points on digital curves", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. II No. 8, 1989, 859-872.