Building and maintaining computer representations of two-dimensional mine maps

John Albert Horst and Tsung-Ming Tsai

Intelligent Systems Division, Manufacturing Engineering Laboratory
National Institute of Standards and Technology (NIST)
Building 220, Room B-124
Gaithersburg, Maryland 20899 USA
{horst, tsung}@cme.nist.gov

Abstract

We present algorithms that allow a computer-assisted underground coal mining system to do real-time conversion between two representations of spatial occupancy, namely, certainty grids and object boundary curves. These algorithms can be used to accomplish many tasks of an underground mining vehicle such as mapping, navigation, object recognition, vehicle localization, and vehicle motion simulation. For conversion from certainty grid to object boundaries, an edge linking algorithm [1] is appropriately modified. Certainty grid 'images' are transformed into a 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 certainty grid to object boundary algorithm is tested against simulated noisy certainty grid maps. Finally, an algorithm to convert object boundary curves to an occupancy grid is presented.

1. Introduction

1.1 Background

For several years the Intelligent Systems Division (ISD) of the National Institute of Standards and Technology (NIST) has been supporting the Pittsburgh Research Center (PRC) of the United States Bureau of Mines (USBM) in their development of prototype computer-assisted coal mining systems and sensors. Not surprisingly, such systems are very complex. In order to remove human workers from the coal face [2], a large suite of operations (including coal cutting, haulage, ventilation, and roof bolting involving several pieces of machinery) must be precisely coordinated and synchronized.

NIST ISD has developed a reference model architecture called the Real-time Control System (RCS) architecture and methodology [3]. RCS is specifically designed to handle complex control systems applications. Additionally, RCS focuses on problems with a preponderance of synchronization and coordination operations (as opposed to problems that can be completely

described through mathematical analysis[1]). RCS has been applied successfully to the development of prototype computer-controlled coal mining systems by the USBM [4].

The work reported herein, namely, mine map conversion algorithms, was inspired by the unique problems of map generation, use, and maintenance provided by the coal mining automation problem. One unique aspect of a coal mine (different from most mapping problems in the mobile robot literature) is that the space (the mine) is known to be completely occupied before cutting begins, i.e., it is completely occupied (with coal). Therefore, the cutting operation itself will give most of the needed information on spatial occupancy in the mine, i.e., where coal is removed, we have empty space; where coal remains (or where there is gob[2]), we have occupied space.

1.2 Mapping spatial occupancy in a mine

Maps can contain a large and varied amount of information. In a coal mine we are particularly interested in spatial occupancy maps, i.e., maps which indicate where there is coal (or gob[2]) and where there is no coal. We also need mine maps to indicate various landmarks, properties or objects of importance, for example, the location of man doors in the mine, floor conditions, landmarks, roof conditions, and coal composition. Such maps are often called 'annotated' maps. This work focuses on maps of spatial occupancy only, although adding annotations is relatively straightforward. Henceforth, when we refer to a 'map', we mean a 'spatial occupancy map'.

Our goal is to have a computer-controlled mining machine follow a conventional mine plan and build/maintain a mine map as it extracts coal while following this mine plan. Underground coal mine maps can be effectively represented as two dimensional (2D) maps and typically are. Therefore, we will deal with 2D maps only, however, 3D or 2-1/2D representations[3] may eventually be needed. While following the mine plan, building and maintaining occupancy maps needs to be done using cutting history along with machine position and orientation sensors and/or range sensors. All this must be represented and utilized in machine format. Computer-controlled mining systems require several types of mine maps:

- a map that represents the cutting plan (or mine preplan).
- a map of what was actually cut (a 'measured' map), containing the computer's current best estimate of the spatial occupancy of the mine.
- a global map of the whole mine in some data reduced form, perhaps at low resolution
- local maps, perhaps at high resolution.

It is useful to build and maintain a measured map of the mine, since the measured map will deviate to some degree from the original plan and is needed for mobile mining machine navigation. The measured map must be represented within the computer in a form suitable to the tasks performed by the mining equipment to be controlled.

---

[1] Differential and difference equations.

[2] The gob is that part of a mine which is effectively occupied, but from which the coal has been extracted.

[3] An example of a 2-1/2D representation is to combine occupancy information with coal seam height at each occupancy grid location.

1.3 Multiple map representations

In order to perform a wide variety of tasks, intelligent control systems requiring maps are well served by as many map 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 is becoming more and more feasible. Two useful and complementary map representations are certainty grids [5] and object boundary curves[4]. They are particularly useful for a vehicle in an underground mine. For example, for continuous mining machine motion simulation, it is important to obtain vectors normal to object boundaries [6]. Vectors normal to an object boundary are difficult to get from a certainty grid, but are relatively easy to obtain from an object boundary curve. Spatial occupancy information is gotten easily from certainty grids but not as easily from object boundary curves. Additionally, representing spatial occupancy in the form of object boundary curves allows us to detect higher level features such as corners, curves, and lines. As a result, high level geometric features are easier to compute, perceive, and update. For example, this information can be used by a mining vehicle to reorient itself. This is particularly important when hierarchical control architectures are used [3]. It is relatively easy to build and maintain a certainty grid, making 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.

Since we want to maintain simultaneously these two representations of a dynamic occupancy map, we need real-time conversion algorithms. The certainty grid to object boundary (CGOB) algorithm we have developed has been tested against certainty grids of various types corrupted by blurring and speckle noise (as in Figure 6). The object boundary curve points encode (in the ordering of those points) which side of the curve is occupied.

The CGOB algorithm concludes with a piecewise linear curve approximation algorithm. The popular split and merge approach [7, 8] is known to be inefficient and several attempts to improve its efficiency result in an increase in complexity [1, 9]. We have developed a new approach that is accurate, simple, and efficient. We compare this new approach to others in the literature.

The second algorithm we present is an object boundary to occupancy grid (OBOG) algorithm which has been tested against object boundary curves of various types (Figure 9 and Figure 10). This algorithm is significantly simpler than the CGOB algorithm and can also be computed in parallel on a pixel processor.

In order to be of use in a large-scale real-time intelligent control system, these algorithms need to fit logically into a consistent architecture for real-time intelligent control. Hierarchical control (such as RCS [3]) specifies 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 CGOB and OBOG algorithms will allow both representations to simultaneously exist and be updated in a real-time intelligent system as illustrated in Figure 1. In Figure 1, sensory processing levels are represented as SP1 and SP2; world modeling levels are represented as WM1 and WM2.

---

[4] Object boundary curves are basically equivalent to 'polygonal maps', the latter being terminology common in the mobile robot path planning literature.

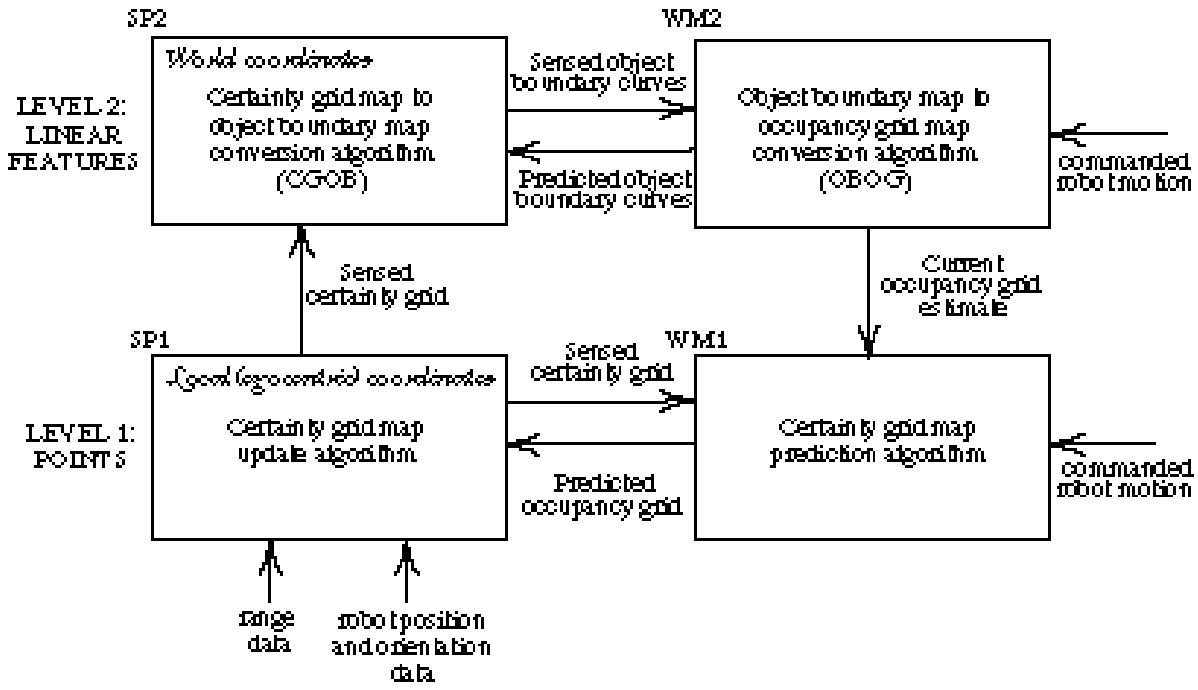

Figure 1: Sensory processing and world modeling components of an intelligent system showing functionality of map conversion algorithms.

2. The certainty grid to object boundary algorithm

We argued in Section 1.3 that both certainty grid and object boundary curve maps of spatial occupancy are very useful for common tasks of a mining machine in an underground coal mine. Since we have two representations of the same space, we need real-time conversion algorithms to maintain the consistency between the two representations. We now describe the CGOB algorithm, which converts from a certainty grid into a set of object boundary curves[5]:

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 quantization noise).
5) Approximate the smoothed boundary points with contiguous line segments by monitoring change in chord length and path length.

[5] This algorithm is a modified form of that found in [1]. Unless explicitly stated, no attempt is made to differentiate those parts of the algorithm that have been modified and those that are not (piecewise linear curve approximation is the most significant modification).

2.1 Edge detection

Nevatia & Babu [1] used six 5x5 masks as 'compass' operators [8] (*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.

Edge operators of size 3x3 didn't produce smooth thinned edges, so 5x5 operators were required. The 5x5 operators we used were called stochastic gradient operators [8]. 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 1.0.

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 2. Edge thinning is accomplished by requiring the following for a cell to be an edge cell:

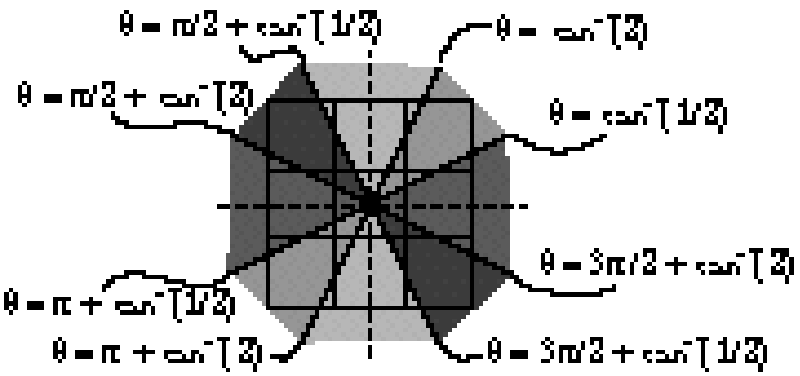

Figure 2: Regions of edge orientation in the neighborhood of a cell.

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

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 cell in the thinned edge grid are chosen as follows:

1) Determine within which of the eight regions (illustrated in Figure 2) the edge orientation lies. Edge direction is defined $\pi/2$ counterclockwise from the edge gradient direction. This means that occupied space is always to the right if moving along the edge in the edge direction.
2) 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), 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 as in dynamic programming techniques [10, 8], 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 of size five 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.

2.3.2 Curve generation

The grids of successors, predecessors, and forks, are used 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 as starters 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 using splines and higher order polynomials, but line segments have the advantage of simplicity. We avoided optimal linear approximations [11] since they are computationally much more expensive [12].

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' [13, 7, 8]. It uses a minimum mean squared error criterion. However, the split and merge method has some essential weaknesses:

1) It is inefficient since 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 [1].
2) Errors can occur in the choice of the initial break point [9].
3) Attempts to eliminate these problems come with a substantial increase in complexity [9, 1].

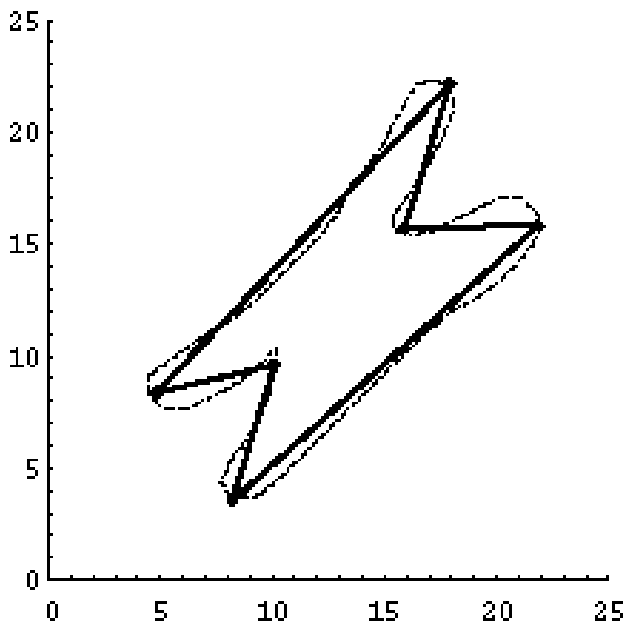

Figure 3: A Gaussian smoothed (window size = 5) digital closed curve with fitting by chord and arc length with least squares fitting.

The strengths of the split and merge algorithm are:

1) Simplicity.
2) It is controlled by a single, physically meaningful parameter (maximum deviation).
3) It generates a sparse set of points that well approximate the original curve.
4) The approximations are stable (according to the criteria defined in [14]).
5) It preserves symmetry.

We sought an algorithm that is efficient and minimizes mean squared error, but which preserves the strengths of the split and merge approach. Such requirements are largely satisfied by monitoring the relationship of chord length and arc length along the curve. However, symmetry and stability are to a small extent traded off for the gain in efficiency.

Following is a description of the chord and arc length method for piecewise linear curve approximation:

1) Determine whether the curve is open or closed.
2) Do local Gaussian smoothing on the raw data in a 'scan-along' manner 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 because the thinning algorithm often produces thinned edge pixels having one four-neighbor and one eight-neighbor (see Figure 8).

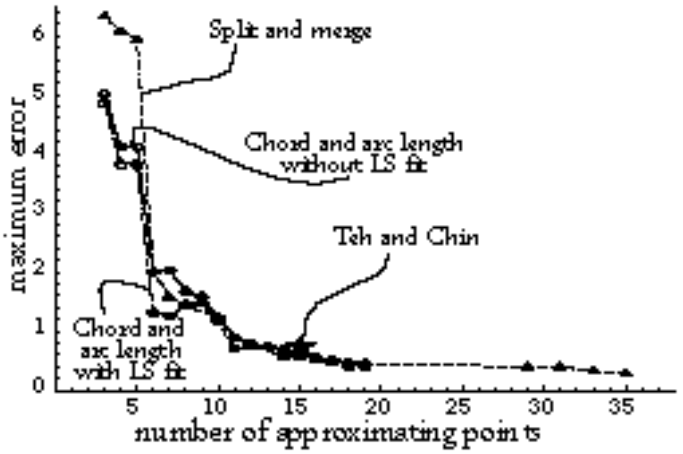

Figure 4: Maximum errors as a function of the number of approximating points for several algorithms using the digital (unsmoothed) curve of Figure 3.

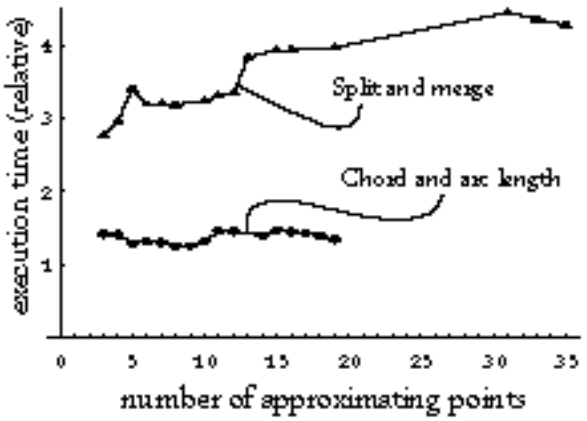

Figure 5: Relative execution time as a function of the number of approximating points for the chord and arc length algorithm and the split and merge method using the curve of Figure 3.

We illustrate in Figure 4 the maximum error of the various algorithms for the digital closed 'chromosome' curve used in Figure 3. Note that as the number of approximating points increases, the maximum error for the split and merge and the chord and arc length methods converge. Figure 5 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).

3. The object boundary to occupancy grid algorithm

The OBOG algorithm we now describe is significantly simpler to state and compute than the CGOB algorithm. This is not surprising, since object boundaries are a more compact representation than certainty grids and can be considered to be at a higher level in the hierarchy of the intelligent system. This is because object boundary curves are beginning to organize space into lines whereas certainty grids organize space into disconnected points. More work is required to discern this connectivity of points into lines and then to make appropriate approximation of those lines. Input to the OBOG algorithm are the object boundary curves which define spatial occupancy. Each of these curves is oriented so that as one moves along the curve, occupied space is to the right. For each cell in the grid:

1) Find the point on the set of boundary curves that is closest to the cell. This will either be an endpoint of two contiguous line segments or a point within a single line segment. If there are two or more points on the curves that are at the same distance from the cell, one can be picked randomly without error.

2) If the point on the curves closest to the cell is a point within a line segment, compute the cross product of the vector from the cell to the point and the vector of the (oriented) line segment. If the sign of the cross product is positive, the cell is in occupied space; if negative, the cell is in unoccupied space.
3) If the point (on the curves) closest to the cell is an endpoint of two contiguous line segments, form the dot product of the vectors formed by the two contiguous (oriented) line segments.
4) If the sign of the dot product is positive, compute the cross product of the vector from the cell to the endpoint and the vector of either of the (oriented) line segments. If the sign of the cross product is positive, the cell is in occupied space; if negative, the cell is in unoccupied space.
5) If the dot product is less than or equal to zero, form two cross products of the following pairs of vectors a) the vector from the cell to the endpoint and the vector of the first of the two (oriented) line segments and b) the vector from the cell to the endpoint and the vector of the second of the two (oriented) line segments.
6) If these cross products both have positive sign, the cell is in occupied space. If both have negative sign, the cell is in unoccupied space.
7) If these cross products have different signs, form the vector of the sum of two unit vectors in the direction of the vectors formed by the two line segments on the curve. Compute the cross product of the vector from the cell to the endpoint and this sum vector. The cell is in occupied space if this sign is positive and is in unoccupied space otherwise.

4. Experiments

For testing the CGOB algorithm, we used simulated grid maps with added Gaussian blurring and speckle noise. Figure 6 shows an example of such a noisy grid. Note that the final approximating line segments are superimposed. Here we see that both open or closed curves can be detected and that data reduction from the certainty grid to the obstacle boundary points is by a factor of about 145 in this example. Figure 7 and Figure 8 show the raw edge grid and the thinned edge grid. Programming and testing was done in both *Mathematica*™[6] and 'C'.

For testing the OBOG algorithm, we used object boundary maps like those in Figure 9 and Figure 10. Note that the initial line segments constituting the object boundaries are superimposed over the resultant occupancy grid. Programming and testing was done in *Mathematica*™.

5. Conclusion

Some of the uses and advantages of having the CGOB and OBOG algorithms in a real-time computer-controlled mining system are:

1) The 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

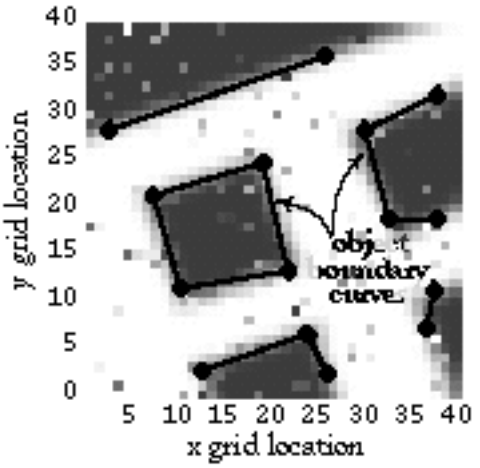

Figure 6: A noisy certainty grid with final approximating points superimposed.

[6] Reference to product or company names are for identification only and do not imply NIST endorsement.

former for local, egocentric maps and the latter for global maps. Using the OBOG algorithm, the system can bring in sections of the global map (as obstacle boundary curves) and convert them into certainty grids, as needed.

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.

We have shown that the chord and arc length algorithm used for linear approximation has excellent error performance, simplicity, and efficiency with only a small sacrifice in stability and

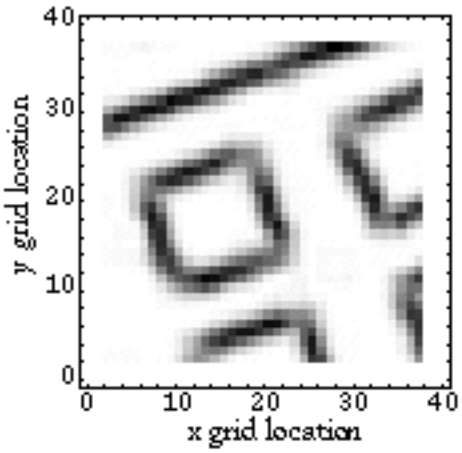

Figure 7: The raw edge grid for noisy certainty grid of Figure 6.

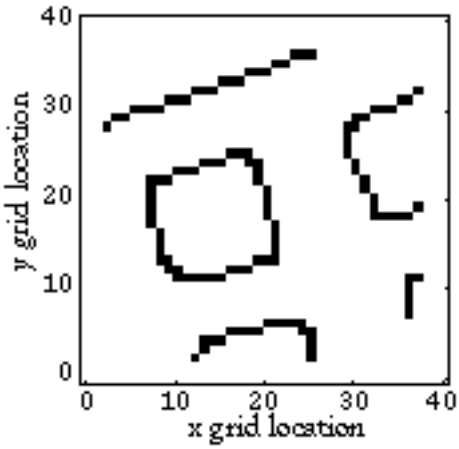

Figure 8: The thinned edge grid for noisy edge grid of Figure 7.

symmetry compared to other approaches in the literature.

More work is required in at least two areas. 1) We need to convert the algorithms to represent three dimensional space. 2) Maintaining the certainty grid as a local map and obstacle boundary curves as a global map requires that there be a way to integrate a local obstacle boundary map into the global obstacle boundary map. Currently, the certainty grid and the obstacle boundary curves represent one map, namely, the global map.

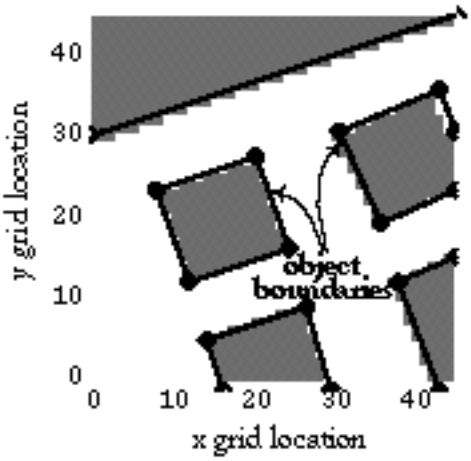

Figure 9: An example set of object boundary curves with the resultant occupancy grid overlaid

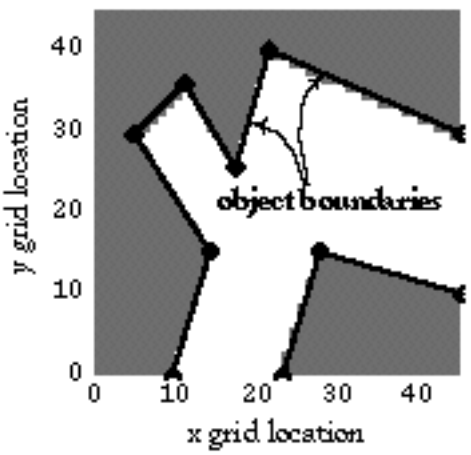

Figure 10: Another example set of object boundary curves with the resultant occupancy grid overlaid

We hope to use the CGOB and OBOG algorithms in a variety of real-time mobile robotic systems, though it is particularly suited for the unique mapping problems found in an underground coal mine. The most computationally expensive portions of the CGOB algorithm are edge detection, thresholding, thinning, and the generation of predecessors and successors. These portions are designed to be computed on image processing hardware. Similarly, the entire OBOG algorithm can be computed using image processing hardware. We are confident that this will allow the CGOB algorithm to operate fast enough for real-time mobility tasks.

References

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

[2] McClelland, John J., R.F. Randolph, G.H. Schnakenberg, Jr., and R. S. Fowkes. "Reduced Exposure Mining Systems (REMS)", Bureau of Mines Special Publication SP-26-94

[3] Albus, J. S., "Outline for a Theory of Intelligence", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 3, May/June 1991.

[4] Schnakenberg, George H., Jr., "Progress Toward a Computer Operated Mining System", Proceedings 10th International Conference on Coal Research, Coal: Energy for the Future, October 9-12, 1994, Brisbane, Queensland, Australia, Vol. 1, pp. 475-486.

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

[6] Horst, John A., "Environment simulation for a continuous mining machine", Final research agreement report from NIST to the U.S. Bureau of Mines, Agreement J0389338, February, 1994.

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

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

[9] 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.

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

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

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

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

[14] Fischler, Martin A. and Bolles, Robert C., "Perceptual organization and curve partitioning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8 No. 1, January 1986, 100-105.