4-2004

Cognitive Robot Mapping with Polylines and an Absolute Space Representation

Kennard R. Laviers

Gilbert L. Peterson
Air Force Institute of Technology

Recommended Citation

Laviers, K. R., & Peterson, G. L. (2004, April). Cognitive robot mapping with polylines and an absolute space representation. IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004. https://doi.org/10.1109/robot.2004.1308856

Cognitive Robot Mapping with Polylines and an Absolute Space Representation

Kennard R. Laviers and Gilbert L. Peterson
Department of Electrical and Computer Engineering
School of Engineering and Management
Air Force Institute of Technology
Wright-Patterson, AFB, 45433-7765
[kennard.laviers, gilbert.peterson]@afit.edu

***Abstract*— Robot mapping even today is one of the most challenging problems in robot programming. Most successful methods use some form of occupancy grid to represent a mapped region. This approach becomes problematic if the robot is mapping a large environment, the map quickly becomes too large for processing and storage. Rather than storing the map as an occupancy grid, our robot (equipped with sonars) sees the world as a series of connected spaces. These spaces are initially mapped as an occupancy grid in a room by room fashion. As the robot leaves a space, denoted by passing through a doorway, the grids are converted to a polygonal representation. This polygonal representation is stored as rooms and hallways as a set of Absolute Space Representations (ASRs) representing the space connections. Using this representation makes navigation and localization easier for the robot to process.**

I. Introduction

Learning a map of its environment presents a complex problem for robots. The problem begins with getting the robot to move in such a fashion that it explores the entire environment, insuring a complete map is constructed. While the robot explores and records real time sensor data, it uses that data (range and motor data) to form a map of the explored region. These maps allow the robot to develop a sense of a place and localize themselves in the world. Memory space limitations require creative solutions for the representation of the map itself when mapping large environments. These maps can take on a number of internal representations from a grid to a general graph of interconnected spaces; our method combines the best of both approaches, allowing for greater efficiency performing planning and localization.

An associated problem to learning the map includes localization of the robot. Accurate localization is a prerequisite for building a good map and having an accurate map is essential for good localization. Simultaneous Localization And Mapping (SLAM) is a critical underlying factor for any successful mobile robot navigation.

A. Mapping

The representation of the map is a key factor of robot mapping. Implementations of robot mapping represent map data as either an occupancy grid [14][2] or a topological network [12][11][7]. An occupancy grid is a matrix of cells where cells containing a value greater than 0.0 indicates a belief that an obstacle resides in space represented by the cell. Using a matrix in this fashion requires the reservation of a large amount of memory for representing the entire maximum map size. A topological map is one in which objects are stored with respect to each other based on a *robot centric* point of view. This representation makes it easy to use memory only as needed, generating more efficient maps, but introduces implementation difficulties and cannot easily convert to human-understandable maps.

Much like a graph with nodes and edges as paths between locations, our method represents the map as a set of line segments or polylines. This data representation significantly reduces the size of the data structure. This representation also lends itself to faster path planning and a more human centric representation. However, generating the polygonal representation with sonars is difficult since the noise generated by the sonars significantly complicates the process.

To overcome the issue of sonar noise, we start by creating a histogram map or occupancy grid from the sonar input/data. Using a new technique of filtering the data in the histogram, which reduces the histogram wall thickness to one, the data is cleaned and then converted to a list of vertices using the Douglas Peucker line reduction algorithm [5]. The remaining set of lines represents the original map of the room. These lines, along with other topological information, are stored in an Absolute Space Representation (ASR) data structure and used later to form a complete map. This process repeats for each room or space visited.

B. Localization

Localization is the process of correcting errors in the robots *dead reckoning* system, the sensors that measure detailed movement of the robots. There are broad categories of localization: local and global. Local techniques compensate for odometer errors and require that the initial location of the robot is approximately known. Global techniques can localize a robot without any prior knowledge about its position; and they can handle the *kidnapped robot problem*, where a robot is kidnapped and carried to some unknown location. Obviously, global localization techniques are stronger than local ones and can deal with situations in which the robot is likely to experience serious positioning errors [15]. While our system

is being extended to perform localization, it is not a part of this discussion.

In the next section we cover key areas of research that are building blocks in our work. In section IIa, we explore obstacle mapping, and focus on Borenstein and Koren's technique [2] and how we extend it in our work. In IIb, some cognitive representations of the robot maps are discussed with the focus on the use of ASRs. Section III shows the development of these idea into an implementation. Results of simulated and real world mapping are shown in section IV. Finally, future work and extensions are discussed in section V.

II. Related Work

The goal of robot mapping is to build a useful map and provide a solid method for the robot to identify and avoid obstacles in its path, and if revisiting a location to recognize that fact. While the robot explores the environment, it records data readings from sensors and stores that data for this purpose. In the following subsections we discuss the two main approaches of maintaining and representing this information. In section IIa, we explore obstacle mapping and how it relates to our work. Section IIb covers cognitive mapping used with polylines space/room fragmentation.

A. Occupancy Grid

One of the most straight forward methods of representing a map has been the occupancy grid [14]. In this method, the world is represented as a two dimensional array of probability information about the occupancy of a grid cell. Using information from the robot's range sensors and pose information, an array cell is updated with new probability information after each sensing action [14]. Borenstein and Koren extend the occupancy grid map idea allowing for faster processing and making it possible for use in real time obstacle avoidance and robot navigation by approximating the probability values [2].

Borenstein and Koren's approach, Histogram In Motion Mapping (HIMM) [2], represents obstacle data in a two dimensional array denoted as C. When a sonar indicates a reading for a specific cell C_{xy} in the array, C_{xy} is modified so that $C'_{xy} \leftarrow C_{xy}+3$ and all its surrounding cells $(x\pm1)(y\pm1)$ are incremented by 1.5. All cell values in C are limited so that $0 \leq C'_{xy} \leq 15$. Additionally, cells along the path from the robot center point (x_0, y_0) to (x, y) denoted as C_{ij} are decreased by 1. Notice that it assumes a higher belief that a cell is occupied than not occupied. This simple method of building a map demands little processor time and memory space, so is quickly processed by a computer.

Borenstein and Koren also introduce the idea of using a polar histogram for obstacle avoidance [3]. Within a predetermined window around the robot, a polar sweep is made in 5° increments. The angle β of the current point (i, j) with respect to the vehicle center point (x_0, y_0), is determined by,

$$\beta_{i,j} = arctan\left(\frac{j - y_0}{i - x_0}\right) \tag{1}$$

For each cell (i, j), magnitude is calculated by,

$$m_{ij} = C_{i,j}^2 * s \tag{2}$$

and s is set so that the value within C_{ij} is scaled where the closer it is to the robot the higher the value it is assigned, and the farther away, the less weight it is given. For each cone ϕ, cells are summed to determine a magnitude vector $\vec{k}_\phi$.

$$\vec{k}_\phi \leftarrow \sum_{ij} m_{ij} \tag{3}$$

Determination of which cone point (i, j) belongs to is accomplished by finding β_{ij}, where $\phi - 1 < \beta_{ij} \leq \phi$ indicates point $(i, j) \in$ the cone from $(\phi - 1, \phi]$. The robot explores in the direction of lower values of $\vec{k}$. An advantage of HIMM is its ability to progressively adapt the strength of an obstacle avoidance reaction to the level of evidence for the existence of an obstacle, and do so quickly. By using this technique which is computationally cheap, we free up processor time for use in the cognitive conversion process while providing obstacle avoidance.

Another popular approach to mapping, known as Expectation Maximization (EM)[6], holds probabilistic information about the world and constructs the most likely map from it.

B. Cognitive Mapping

Using a simple occupancy grid presents several problems. Of these problems, the amount of memory needed to represent a large area is of primary concern. For example, if the grid resolution is 10 centimeter by 10 centimeter cells, mapping a 100×100 meter area requires $(100 \times 100) \times (100 \times 100) \times 16_{bytes} = 1.53_{GB}$ of memory space.

Cognitive mapping with the use of ASRs is a topological mapping approach. That is, the world is represented more from the perspective of the robot and in some cases maintains very little metric information about the world. Instead, the map is created with consideration to where the ASRs are with respect to each other and the robot.

1) Absolute Space Representation: An Absolute Space Representation or ASR [7], is a cognitive mapping technique used to build models of rooms or spaces visited. Absolute space comes from the idea that the representation for each space should be independent of all other spaces. Also, humans view the world cognitively and so should our robots.

Rooms are separated by access points, represented as edges connecting nodes. This technique is explained by Hill, Han, and Lent [7]. While the idea of the ASR [7] has a foundation with three dimensional image based localization and mapping (much like VSLAM [15]), it is also applicable with respect to indoor mapping using ultrasonic sonars as the range device [8]. Figure 1 shows a standard Cartesian map translated to an ASR. Notice the rooms become nodes and the exits become edges connecting the nodes. The use of ASRs becomes more necessary as the size of the mapped environment increases. By placing non-current ASR data in persistent storage, the robot must only hold a limited amount of information in main memory at any given time. Additionally, when the robot uses

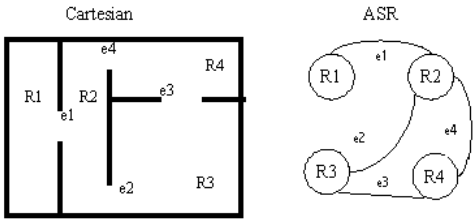


Fig. 1. Two Room ASR Example

the cognitive map at a later time, it need only know which ASR it currently occupies and have only the data for that ASR, and its neighboring ASRs loaded into memory.

Jeffries and Yeap [12] expand this theory of cognitive maps which at its core has the notion of an autonomous agent building a map in the memory, i.e. a *map in the head* termed a *cognitive map*, for the places it visits. The agent must first develop a representation for each space visited [19][18]. The space occupied by the robot, termed the *local space*, is defined as the region which the robot perceives enclosing it (a room; hall, etc). The robot's cognitive map grows as the representation of each local space it visits is added to a topological network or graph of ASRs. They show how this theory is applied to the problem of an autonomous mobile robot equipped with sonar sensors, building a map from its experience of the places it has visited [12]. In this case the cognitive map consists of a *local space representation* for each local space visited with connections to other rooms that have been experienced as neighbors. All local space representations have their own local coordinate systems and are independent of all others. Our approach builds on this method by applying the use of polylines to the representation of the ASR. This enhances our ability to use the ASR for navigation and localization.

2) Polyline Simplification: Most implementations of robot mapping represent map data as some form of occupancy grid or topological network. Our approach merges the two ideas. We maintain world data by transforming a temporary grid to a set of lines and structuring those lines in a topological network. The size of the data structure is significantly smaller using polylines instead of a grid, and using the data for path planning and localization becomes much faster as the search space is reduced. Using a polygonal representation with sonars has been previously considered infeasible [9] as the amount of noise generated by the sonars complicates the process greatly. Latombe and Banos [9] use polylines with the help of a laser range finder, instead of sonars. The vehicle moves from point to point and performs a polar sweep at each point. Each new polar sweep is added to the current model. Lines are extracted from the raw range data using a polar line fitting algorithm. Using sonars necessitates the need for developing a new method of fitting range data to lines. We extend their system by developing a new data fitting method and storing the data in a cognitive map for use in later localization and navigation.

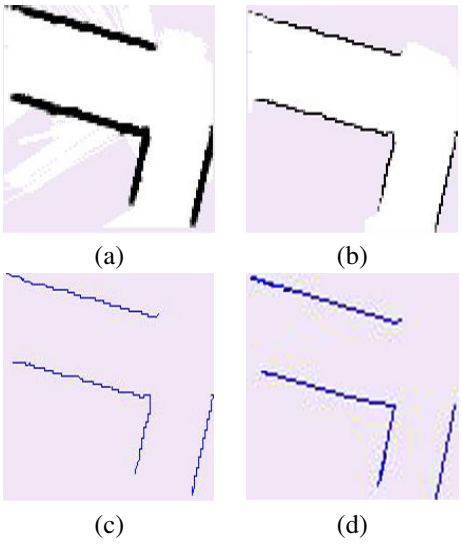


Fig. 2. Complete polyline process. The raw map is shown in a (50X50=2500 integer matrix) (a). followed by the cleaned map(50X50 integer matrix) (b). Figure (c) is the raw polyline (250X2=500 integers). Finally (d) is the simplified fitted polyline (9X2=18 integers).

III. Implementation

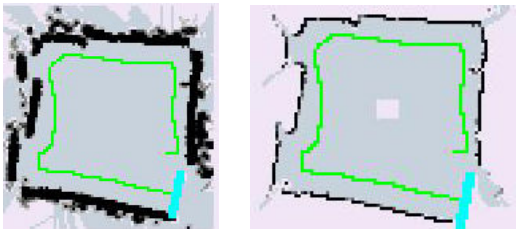
Our system is comprised of several distinct algorithms. At the highest level, the system consists of the robot traveling around its environment creating a HIMM map. Each time the robot detects that it leaves a room (enclosed area), it creates a complete polygonal map of the previous room and saves that map to the hard drive as ASR to cartesian map information. A new map is started and the process begins again in the new room.

Creating a map of a room is a multistage process. It begins by cleaning the HIMM map (Fig. 2a) leaving walls represented by no more than one cell thick lines in the grid (Fig. 2b). Next, an extraction algorithm is called to create an ordered list of vertices (Fig. 2c) using a nearest neighbor, follow-and-remove process. Using the new list of vertices, a final representation (Fig. 2d) is generated using vertex and edge reduction algorithms. Exit and entry locations are stored for later use as pivot points for rotational adjustments to compensate for localization errors. Finally, the robot combines all the ASRs to create a human readable master map.

A. ASR Construction

When the robot completes the mapping of one room/space it constructs an ASR from the HIMM map. This construction involves taking the raw HIMM map, cleaning it, and converting it to a cognitive map. Next, the robot adds the map to the ASR container. We break this process up into the sequence of steps which consist of

1) cleaning the raw HIMM,
2) creating an ordered list of vertices by extracting points,
3) eliminating points that are close together,
4) reducing the number of edges,
5) calculating a reference point used later when all the ASRs are put back together,



(a) (b)

Fig. 3. A HIMM before (a) and after (b) applying the HIMM cleaning algorithm. The line inside the room represents the path of the robot.

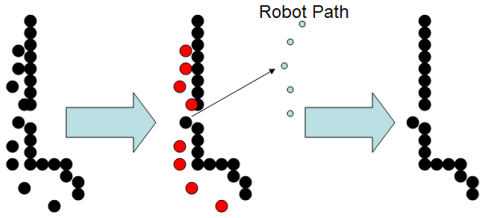


Fig. 4. Any occupied cell not able to draw a ray to the path that the robot traveled without going through another occupied is discounted as noise.

6) and finally adding the new ASR to the ASR container.

B. Clean Map Algorithm

From the HIMM built occupancy grid map, our cleanup algorithm incorporates some modifications of the original mapping technique provided by [2]. In their paper, empty cells $empty(C_{xy})$ are decreased by -1 but not allowed to drop below 0. To increase the difference between occupied and unoccupied, the lower bound of C is decreased so that $empty(C_{xy}) \geq -3$ allowing us to distinguish between an unexplored region and an empty region. More importantly, spaces occupied by the robot itself are marked as $robot_{path}$ for use in the map cleaning algorithm. This is shown in (Fig. 3a) and the modified map demonstrates a cleanly mapped region inside the empty space and messy area in the unexplored region (Fig. 3). The cleaning process is straight forward; any cell deemed empty or occupied is checked to see if a ray from it to the path of the robot is possible. If the ray traverses occupied cells then it is set to an unexplored or unknown state (Fig. 4).

We represent the map to be cleaned as a matrix H where $H_{x,y}$ is a specific position of the map. The final filtered map is H'. Let max_{dist} be the maximum effective range of the sonars. Part of our algorithm requires creating a ray from every point (x, y) where $H_{x,y} > 0$ to $x \pm max_{dist}, y \pm max_{dist}$. All line functions use Bresenham's line algorithm [4] for efficiency. To represent the line algorithm, we define $\Gamma(p_a, p_b, \{start, next\})$ which returns points along the line from p_a to p_b. Finally, ϵ is a threshold value where $H_{x,y} > \epsilon$ is considered occupied.

C. Point Extraction Algorithm

One of the big challenges in this work is taking point data from the two dimensional array and turning it into a list of organized points. The points must be retrieved in such a fashion that they form cohesive polylines. A point $H_{x,y} > \epsilon$ is added to V starting from point p_s and ending at p_e.

While retrieving points on the map it is important to identify where one polyline ends and another begins. To do this we define a threshold γ that is the maximum allowable distance between two cells containing no occupied spaces.

In general, the algorithm retrieves points by following occupied cells, adding the point to V and removing it from H until $|H| = 0$. This algorithm is detailed in (Fig. 6).

D. Edge Simplification Algorithm (Douglas-Peucker)

Before we perform any edge simplification, it is important to reduce unnecessary vertices first. The process of simplifying the list of vertices **V** extracted from the map begins by eliminating clusters of closely grouped points. A tolerance

```
CleanMap(H)
  For (every point (x, y) of H)
    If(H_{x,y} > ε)
      p_a ← (x, y)
      found ← false
      For (−max_dist to +max_dist as j)
        For (−max_dist to +max_dist as i)
          p_b ← ((x + i), (y + j))
          p_t ← Γ(p_a, p_b, start)
          While (p_t ≠ p_b)
            p_t ← Γ(p_t, p_b, next)
            If(H_{p_t} = robot_path)
              found ← true
              H'_{p_a} ← H_{p_a}
              Break While loop
            If(H_{p_a} > ε)
              Break While loop
          If (found)
            Break For loop
        If (found)
          Break For loop
  End For
  H ← H'
```

Fig. 5. Algorithm to clean a modified HIMM map

```
ExtractPoints(V, H)
  For (0 to |H_y| as j)
    For (0 to |H_x| as i)
      If (Hi, j > ε)
        p_s ← (i, j)
        p_e ← p_s
        j ← |H_y|
        i ← |H_x|
  DO
      V ← V ∪ {p_s}
      H_{p_sx, p_sy} ← 0
      p_s ← Next closest occupied cell
      If (Δ(p_s, p_e) > γ))
        Mark point this as the end of a polyline
      p_e ← p_s
  UNTIL (|H| = 0)
```

Fig. 6. ExtractPoints

Fig. 7. Line fitting before and after.

value τ is defined such that for any point V_i, if $\Delta(\mathbf{V}_i,\mathbf{V}_{i+m}) < \tau$ for $m = \{i, i+1, i+2, i+3, ...n\}$, then $\mathbf{V}_{i+m}$ is removed from $\mathbf{V}$. When $\Delta(\mathbf{V}_i,\mathbf{V}_{i+m}) > \tau$ then i is set $i \leftarrow i+m$.

The Douglas-Peucker algorithm [5] uses the closeness of a vertex to an edge segment as judgment criterion for elimination. It starts with a crude initial guess at a simplified polyline, i.e. the single edge joining the first and last vertices of the polyline. The remaining vertices are tested for closeness to that edge. If these vertices are further than a specified tolerance from the edge, then the furthest vertex from it is added to the simplification. This process creates a new guess for the simplified polyline. Applying this process recursively for each edge until all vertices of the original polyline are within tolerance of the simplification yields the final simplified set of edges.

E. Line Fitting

The final step of the process involves using vertices returned by the Douglas-Peucker algorithm as end points to fit a *linear regression line* [13] of the form $y = b_0 + b_1 x$ using the *least squares* method where

$$b_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{4}$$

$$b_0 = \bar{y} - b_1 \bar{x} \tag{5}$$

Letting $\bar{x}$ and $\bar{y}$ be the averages of the x and y points respectively.

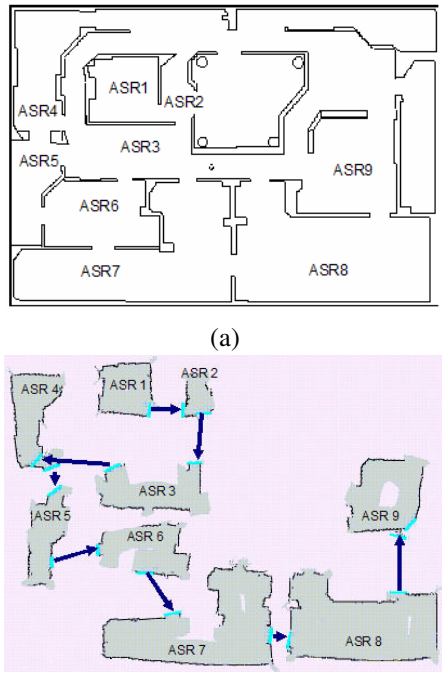
Pearson's correlation coefficient $\hat{\rho}$ is calculated to determine line suitability for a linear fit and if not, reject the fitted line. The coefficient $\hat{\rho}$ is represented as:

$$\hat{\rho} = \frac{n\sum xy - \sum x \sum y}{\sqrt{\left(n\sum x^2 - \left(\sum x\right)^2\right)\left(n\sum y^2 - \left(\sum y\right)^2\right)}} \tag{6}$$

A line with $\hat{\rho} < |.9|$ where $-1 < \hat{\rho} < 1$, is classified as a bad correlation in our implementation. The Douglas-Peucker algorithm restricts this from allowing small noise data to produce large errors in the output map (Fig. 7) requiring this extra fitting process.

IV. Results

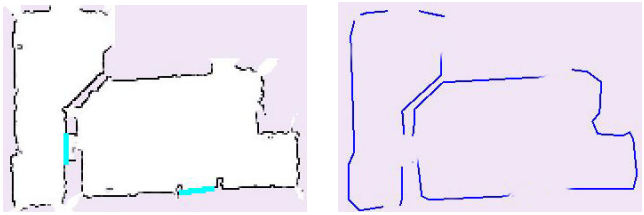
The first series of tests included running the system using the Pioneer robot simulator. The map in (Fig. 8a) shows a map that included 9 ASRs and those rooms represented as a graph (Fig. 8b). In (Fig. 9) an example ASR before and after conversion to its simplified polygonal representation,



(a)

(b)

Fig. 8. ASR's Represented as a Graph. The actual map is shown in (a). The graph (b) shows how the rooms that were mapped translates to a graph.



(a) (b)

Fig. 9. A 2 Room ASR Example showing the polygonal representation (b) of the original HIMM map (a). This figure is rooms 5 and 6 of (Fig. 8a)

mapped from rooms 5 and 6 of (Fig. 8a). The polygon requires less than 100_{Bytes}, while the HIMM used $42K_{Bytes}$ for representing the rooms. Cell size in all results is $10_{cm} \times 10_{cm}$. Additionally, thresholds are set:

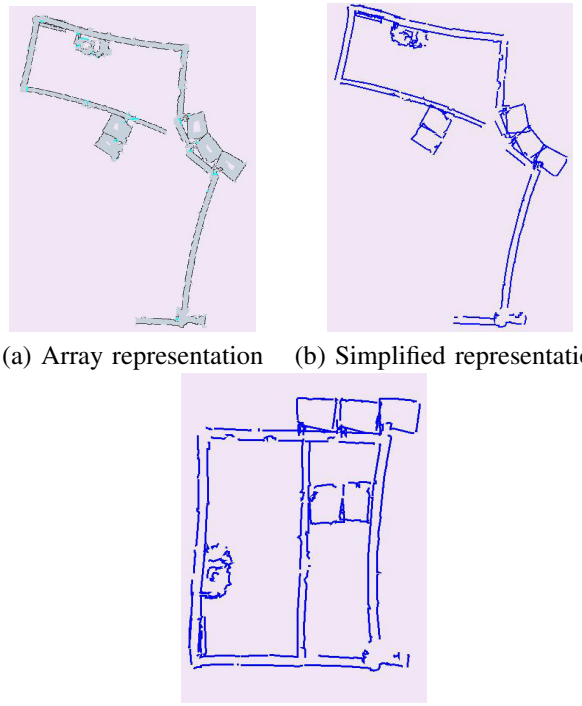
1) $\epsilon \leftarrow 7$,
2) $\tau \leftarrow 2$,
3) and $\gamma \leftarrow 4$.

Using the system in a real world area produced better results. We show in (Fig. 10) a mapping of the first floor of the engineering building at the Air Force Institute of Technology minus some offices with dimensions approximately $(255' \times 255')$. On the left (Fig. 10a), illustrates the original polyline map generated by the system without any localization correction. On the right (Fig. 10b), the polyline before rotational localization errors are manually corrected and (Fig. 10c) is the map after the correction is made.

V. Future Work and Conclusion

While many methods exist for robots to map their environment [2][8][14][12][17], as the size of these environments increases, methods using an occupancy grid start to approach maximum constraints [2][14][17]. Overcoming these constraints remains a challenge even today in computer science. Our two fold method (polylines and ASRs using sonars) [2][9][12] combines the best of two robot mapping philosophies. Our system makes it possible for a robot armed only with sonars to map an arbitrarily large area. It is size scalable, with a non-increasing computational time required to map and a linearly increasing computational time required to reconstruct the ASRs after completion of mapping.

The cognitive mapping process allows robots to easily implement corrections to localization errors using the simplified ASRs instead of grid data. In (Fig. 10), a before and after map is shown with a memory footprint of what the map should look like. Additionally, we plan to reduce the ASR size to a fixed size of $3K_{mm}$ allowing the robot to perform iterative localization using an expectation maximization algorithm similar to Thrun [17], matching the current array window of the robot with the previous ASRs to determine the more probable current pose. This idea is taken from work developed by Schultz and William [16], where range sensor data is divided up into several time slices and comparing them to determine an estimate of pose error. Our idea is to extend this technique to ASRs, fixing the size of the ASR to some small value and comparing them against each other in a similar fashion. Figure 10b is the polygonal representation of the map in (Fig.10a). Manually adjusting rotation error among the ASRs yields (Fig. 10c).



(a) Array representation (b) Simplified representation

(c) Localization errors fixed at the ASR level.

Fig. 10. The polyline map of the first floor of the engineering building at the Air Force Institute of Technology. Figure (b) shows the map with localization errors uncorrected, figure (c) shows the map after rotational localization errors are fixed. This algorithm is currently being developed and the result in (c) are simulated.

Acknowledgment

The authors would like to thank the members of the Air Force Research Lab for their generous contributions to this project.

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

References

[1] J. Borenstein, B. Everett, and L. Feng, *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd. Wellesley, MA, 1996.

[2] J. Borenstein and Y. Koren, *Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance*. IEEE Journal of Robotics and Automation, Vol. 7, No. 4, pp. 535-539, 1991.

[3] J. Borenstein and Y. Koren, *The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots*. IEEE Journal of Robotics and Automation, Vol. 7, No. 4, pp. 278-288, 1991.

[4] N. Cossitt, *Introduction to Bresenham's Line Algorithm Using the SBIT Instruction*. Series 32000 Graphics Note 5, National Semiconductor Application Note 524, 1988.

[5] D. Douglas and T. Peucker, *Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature*. The Canadian Cartographer, Vol. 10(2), pp. 112-122, 1973.

[6] S. Thrun, W. Burgard, D. Fox, and F. Dellaert, *Robust Monte Carlo Localization for Mobile Robots*. Artificial Intelligence, Vol. 128(1-2): pp. 99-141, 2001.

[7] R. Hill, C. Han, and M. Lent, *Applying Perceptually Driven Cognitive Mapping to Virtual Urban Environments*. AI magazine, Vol. 23, 2002.

[8] D. Kortenkamp, *Cognitive maps for mobile robots: A representation for mapping and navigation.*, PhD Thesis, University of Michigan, 1993.

[9] H. Gonzanos and J. Latombe, *Navigation Strategies for Exploring Indoor Environments*. International Journal of Robotics Research, 2001.

[10] M. E. Jefferies and W. K. Yeap, *Neural Network Approaches to Cognitive Mapping*. Artificial Neural Networks and Expert Systems, 1995. In Proceedings of the Second New Zealand International Two-Stream Conference, pp. 75-78, November 1995.

[11] M. E. Jefferies and W. K. Yeap, *The Utility of Global Representations in a Cognitive Map*. In Proceedings of the 2001 Conference on Spatial Information Theory, 2001.

[12] M. E. Jefferies, W. K. Yeap, L. Smith, and D. Fergusen, *Building a Map for Robot Navigation Using a Theory of Cognitive Maps*. In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Marbella, Spain, 2001.

[13] J. S. Milton and J. C. Arnold, "Introduction to Probability and Statistics: Principles and Applications for Engineering and Computing Sciences". McGraw-Hill Primis Custom Publishing, 2002.

[14] H. Moravec, *Sensor Fusion in Certainty Grids for Mobile Robots*. In Sensor Devices and Systems for Robotics, Springer-Verlag, Nato ASI Series, pp. 253-276. 1989.

[15] S. Se, D. Lowe, and J. Little, *Global Localization using Distinctive Visual Features* . International Conference on Intelligent Robots and Systems, 2002.

[16] A. C. Schultz and A. William, *Continuous Localization Using Evidence Grids*. In proc. of the IEEE International Conference on Robotics and Automation, Leuven, Belgium, pp.2833-2839, May 16-21, 1998.

[17] S. Thrun, *Probabilistic Algorithms in Robotics*. AI Magazine, Vol. 21: pp. 93-109, 2000.

[18] W. K. Yeap, *Towards a Computational Theory of Cognitive Maps*. Artificial Intelligence, Vol. 34: 297-360, 1988.

[19] W. K. Yeap and M. E. Jefferies, *Computing a Representation of the Local Environment*. Artificial Intelligence, Vol. 107: 265-301, 1999.