Multi-agent Mapping Using Dynamic Allocation Utilizing a Centralized Storage System

Laura E. Barnes & Richard Garcia
Computer Science Department
University of South Florida
Tampa, FL
{lbarnes3,rdgarcia}@csee.usf.edu

Todd M. Quasny & Larry D. Pyeatt
Computer Science Deapartment
Texas Tech University
Lubbock, TX
{quasny,pyeatt}@cs.ttu.edu

***Abstract*— We present a method for multi-agent coordination in mapping using dynamic allocation in a centralized storage system. Typical multi-agent mapping systems use redundant storage of data to allow every agent in the system to have equivalent knowledge of the world. There are several problems with this implementation including the inefficient use of bandwidth and storage space. Our method increases the utilization of the wireless bandwidth by using a singlecast connection to a central storage location. The storage required on an agent is also significantly decreased due to the removal of redundant data on each agent. This method was achieved through a centralized database by two methods: (i) through local dynamic expansion and decompisition of cells; (ii) through dynamic allocation based on the speed, sensor range of the agent, and the global update rate. The proposed method was implemented and tested in both simulation and on an indoor mobile robot. The results demonstrate that the proposed method performs efficiently in both simulation and on actual agents.**

I. Introduction

A mobile robot requires perception of its local environment for both sensor-based locomotion and for position estimation. Occupancy grids [1], [2], based on ultrasonic range data, provide a robust description of the local environment for locomotion, as a way to construct an internal representation of static environments based on ultrasonic range measurements. This method takes into account the uncertainty of sensory data by working with probabilities or certainty values. The occupancy grid representation can be used directly in robotic planning or navigation [3]. The grids allow the efficient accumulation of small amounts of information from individual sensor readings for increasingly accurate maps of the robot's surroundings.

The restriction to statically sized maps is a common criticism of the occupancy grid approach [4]. Prespecification of the map size poses a significant limitations in mobile robotics. Usually the size of an area to be mapped is unknown or the map will later be expanded, therefore a dynamic approach to occupancy grids is more appropriate.

Multi-agent systems are frequently used to map large areas expediently. The sharing and transfer of data between agents in this type of system is a particular problem of interest. One typical solution to the communication issue is to mutlicast or broadcast data to agents within communication range and have them relay that data to agents out of communication range [5]. This type of method incorporates an unassured delivery system allowing for data loss and corruption of possibly critical information. The method presented within this paper allows for the use of a delivery system such as TCP that will assure all data is recieved and that it is recieved correctly.

In this paper, we describe a method of metric mapping with dynamic space allocation utilizing a central storage system. Dynamic space allocation is achieved by allocating a small amount of memory initially and then expanding as the agent(s) explore new areas. The centralized storage location is a single database which all agents update.

We show that multiple agents can reliably map in an inherently distributed way communicating with a single centralized, global storage unit. Minimal data is stored on the agents and no direct sharing of data is required between the agents.

Two different grid configurations are dicussed. Grid configuration 1, the more simplistic of the two, is achieved through local expansion and decompisition of cells. Grid configuration 2 is achieved through dynamic memory allocation based on the speed of the agent, the range of its sensors, and the global update rate. Although more computationally complex, grid configuration 2 is more scalable to heterogeneous teams.

II. Related Work

A. Occupancy Grids

An occupancy grid provides a way of representing data to be later extracted as useful and informative maps. It is basically a data structure that stores data acquired from different sources such as sensors, cameras, etc. In our case

we use sonar sensors (details are presented in the later sections). The data structure represents, for each point in the environment, the probability that the point is occupied by some object. In the occupancy grid, the state variable $s(C)$ associated with a cell, C, is defined as a discrete random variable with two states: occupied and empty, denoted as OCC and EMP, respectively. These two states are mutually exclusive and exhaustive with the following properties:

$$EMP = \neg OCC \tag{1}$$

$$P[s(C) = OCC] + P[s(C) = EMP] = 1 \tag{2}$$

Equations (1) and (2) associate a probability density function with each cell, describing its state. Thus, the occupancy grid corresponds to a discrete-state, binary, and random field. An occupancy grid is realized by estimating the state of each of its cells, typically using sensor data.

Assuming the robot is required to navigate in an office environment, the cell can be used to encode a number of relevant properties to this environment such as occupancy state (occupied or empty), area classification (corridor, workspace, table, wide area, etc.), width, height, etc. Such general world models are known as Inference Grids. This research deals only with occupancy state.

Occupancy grids, first formulated at CMU [6], have been used extensively for mapping [7], [8], [9], exploration and navigation [10]. Different researchers have implemented their own improved versions of occupancy grids. Examples include place-centric grids [11], histogram grids [12], and response grids [13]. Occupancy grids have also been used for localization [14], [15], [16] and position tracking and estimation [17].

B. Dynamically Expanding Occupancy Grids

The occupancy grid architecture typically uses a static array for representing data [3], [2], [18]. This information is used for mobile robot mapping, localization and navigation. We are interested only in mapping, but the architecture we are proposing can be used for localization and navigation also.

A static array for holding the data is appropriate only if the size of the area to be mapped is known. Figure 1 shows a static sized map with individual cells. This is a disadvantage as it restricts the size of the map and would be difficult to update if there are new areas added to existing regions. In which case, a new map may need to be built of the entire region inclusive of the new areas. With dynamically expanding occupancy grids [4], we are able to build variable sized maps and will also be able to increase the size of the map if new areas are added to the existing map. With dynamically expanding occupancy grids, as the robot explores, the map will grow in a non-rectilinear fashion (as shown in Figure 2). This way, the data is represented as it comes in, in a dynamic way.

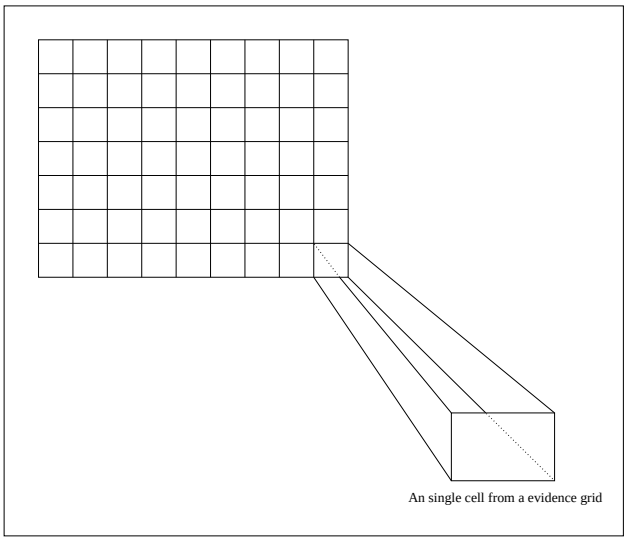

Fig. 1. A traditional occupancy grid

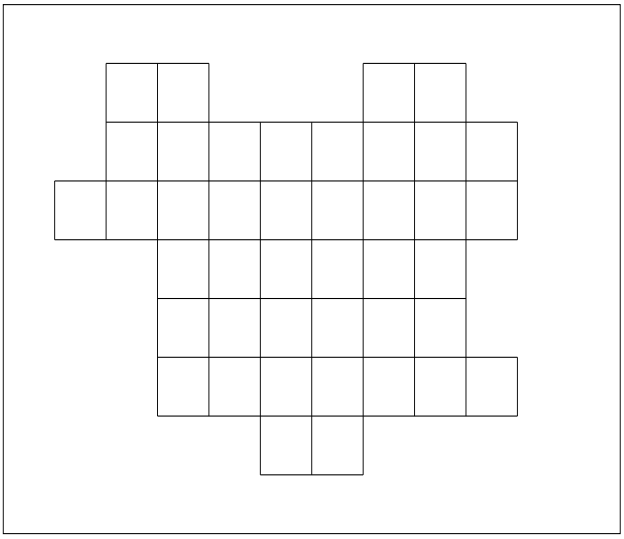

Fig. 2. A dynamically expanding occupancy grid

III. Proposed Method

We present two methods to achieve the desired goal of dynamic space allocation in a centralized storage system. The two methods differ in their dynamic space allocation technique and their global and local update techniques. Results of each method are discussed in section V.

A. Centralized Storage System

The centralized storage system used is a relational database located on a remote machine. The database

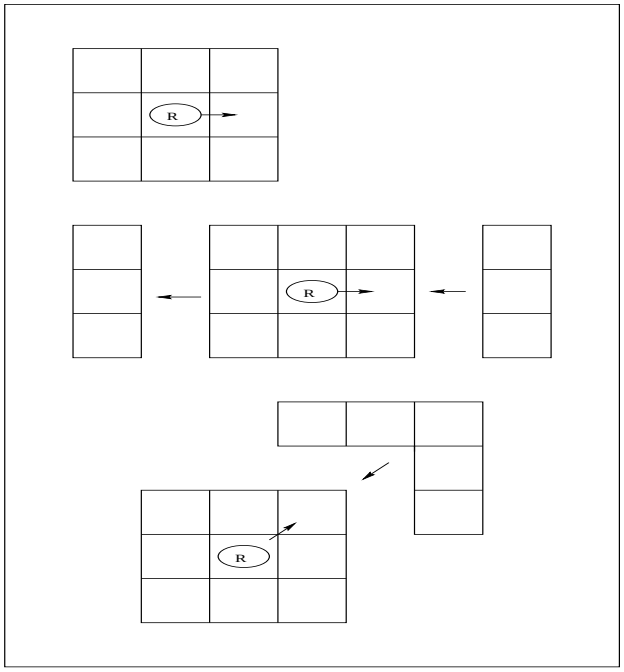

Fig. 3. Adding of new cells to the grid as it expands

contains a 2-dimensional representation of the environment consisting of *x-coordinates* and *y-coordinates* and a probability of occupancy. An index of *x-coordinates* and *y-coordinates* is also used in order to increase the speed of database updates. This technique is valid because there are no redundant *x-coordinates* and *y-coordinates* in the system. Databases are inherently dynamic and therefore were an appropriate storage method. Organizing the data into a relational database also allows for easier readability and comprehension of the collected data. In addition, program maintenance and data sharing are esily facilitated with a database.

B. Grid Configuration 1

With the first grid configuration, mapping is always started at the center of a nine block grid. Each block here is a local map which represents a subset of the global map stored in the database (shown graphically in Figure 3). Each block is a fixed size local map which is divided into cells as a normal occupancy grid. The size of the block and information stored in each cell depends upon the implementation. As the agent explores, new cell probabilities are calculated and periodically updated in the database. Since only a subset of the global map is kept in memory, the map could easily be extended so that a grid cell can store information beyond simple occupancy data. This approach is flexible and imposes no size or shape limits on the global occupancy grid.

The robot is assumed to be in the center block of the

```
Algorithm_Grid ()
{
  initialize_grid ()
  initialize_position ()
  do forever
  {
    get_sensor_data ()
    update_grid ()
    manveur_robot ()
    calculate_position ()

    if new position is
       outside grid block
    {
      add_cells ()
    }
  }
}
```

Fig. 4. Skeleton Algorithm

nine block local grid. Expansion occurs whenever the agent moves out of the center block and into an adjacent block. New blocks are added according to the direction of movement of the agent. If the robot is moving directly into the top, bottom, left or right cell, three cells are added in the direction of movement. If the agent is moving into a diagonally positioned cell, the addition of 5 cells is required in order to keep the nine block grid configuration. Local cell decomposition occurs when cells are added by removing all cells that are nonadjacent to the current central block. The approach is illustrated in Figure 3 and a skeleton algorithm is shown in Figure 4.

Updates to the global map are acieved by passing the *x-coordinates*, the *y-coordinates*, and their associated probability of occupancy periodically to the database. Updates to the global map are achieved by querying the database for the probabilty of occupancy of newly added cells.

C. Grid Configuration 2

For the second grid configuration, space allocation is based upon the agent's sensory range, the agent's rate of movement, and the time between updates. Using these three data components, the maximum amount of potentially viewable space is allocated using the following equation:

$$R = r + (\delta_t) \times (V) \tag{3}$$

In Equation 3, R is the radius of the storage cell, and r refers to the sensory range of the agent. The δ_t value is the time between updates and V refers to the agents rate of movement. Figure 5 is a graphical representation of the space allocation technique.

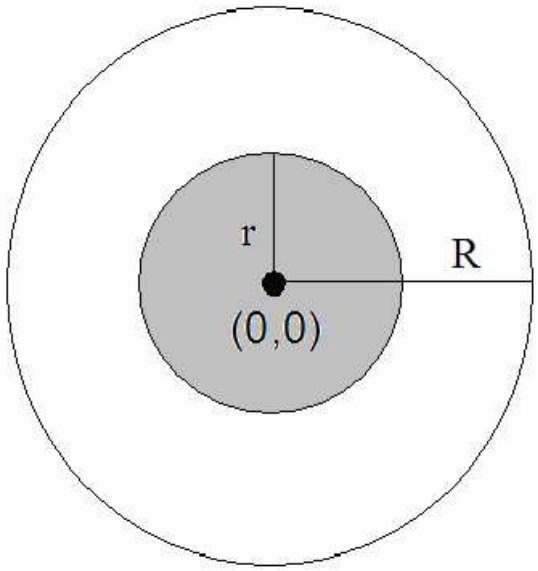

Fig. 5. Grid Configuration 2

Upon the global update the local map is reconfigured to centralize itself around the agent. Updates to the global map are acheived by passing the *x-position* and the *y-position*, the probability of occupancy, and the occupancy grid update equation to the database. Passing the update equation to the database allows for the update to be performed by the central database and removes the necessity of reading from the database and performing the update locally, and then sending the data back to the database.

IV. Implementation

For simulation purposes, the simulator for the Nomad 200 robot is used. Actual tests on the Nomad 200 are also performed. The Nomad Super Scout is an integrated mobile robot system with ultrasonic, tactile and odometry sensing capabilities [19].

For locomotion, the agent is equipped with basic obstacle avoidance and wander behaviors. Dead reckoning is used for the purposes of estimating the robot's *x-position* and *y-position*.

The database is implemented with the open source database management software, MySQL. Any database software would have been sufficient, but MySQL is used because it was readily available freeware and had an available API for the C programming language. The database consists of dynamically expanding records, each containing a primary key, an *x-position*, *y-position*, and a probability of occupancy. The primary key is not actually utilized due to the scope of this project, but good programming techniques for databases require a primary key. The *x-position* and *y-position* are indexed in the database making the combination of the two positions function similarly to a primary key.

For grid configuration 1, the database is initially populated from coordinates $(-150, -150)$ to $(150, 150)$ and sets all probabilities of occupancy for these coordinates to unknown, represented by the value 0.5. The robot is initially positioned at $(0, 0)$. The program also declares a 2-dimensional array of size 300 by 300 which represents the nine initial 100 by 100 grid cells. As the robot's distance from $(0, 0)$ increases, new space is allocated in the database based on the position and direction of movement as described in section 2.2.

Every cell within a 50 inch radius of the robot is updated with every cycle through the sonars. Every one hundred sonar readings, all modified data within the array, is written to the database. All data within the database is read during the initialization phase of the program. The database is also updated and read from when the robot traverses to a new 100 by 100 cell. Grid configuration 1 was tested in both simulation and on the Nomad 200 robot.

Grid configuration 2 initially allocates memory based upon the *R-value* in Equation 3. The rate of movement was sent to a constant value of 1 inch per second and the sensor range was set to 50 inches. Update rates of 1 per sonar cycle and 1 per ten sonar cycles are tested. This configuration was only tested in simulation.

V. Results

Grid configuration 1 was tested with 1, 2, and 3 robots in simulation and on a single Nomad 200. Accurate maps of a hallway setting were made with grid configuration 1. The maps were verified in both simulation and on the actual robot. Figure 6 is the simulation setting used to create the map in Figure 7 with three robots. Figure 9 is a map made with a single Nomad 200 in a corridor environment. Although this configuration provides a mapping method with dynamically expanding occupancy grids, a significant amount of space allocated, in local and centralized storage, is wasted because it is beyond the agents sensory range.

Grid configuration 2 is tested with 1 and 2 robots in simulation. This configuration minimizes wasted storage by calculating the space to be allocated based on specific agent parameters as mathematically represented in Equation 3. The efficiency of allocated space is increased as the number of sonar readings per update is decreases. Figure 10 depicts a map created with one robot.

One noticable limitation of the first method was the approximately 5 sec pause of motion when updating the database. This limitation was less noticable in simulation then on the real robot so further tests and optimization of the grid configuration 1 on the Nomad 200 are needed.

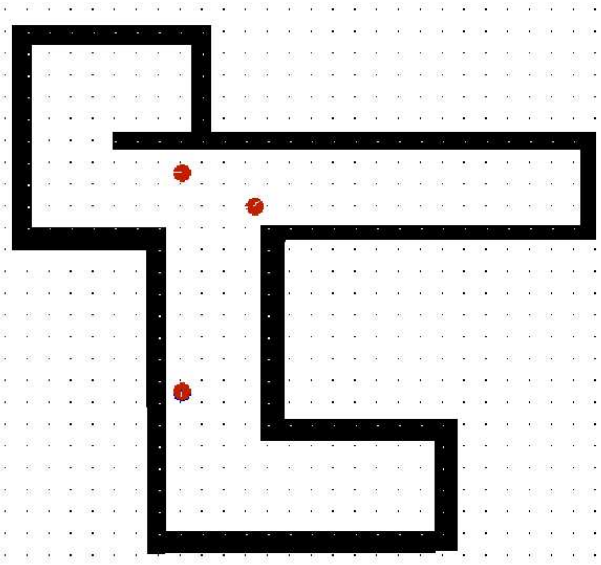

Fig. 6. Simulation Environment Used to Create Map in Figure 7

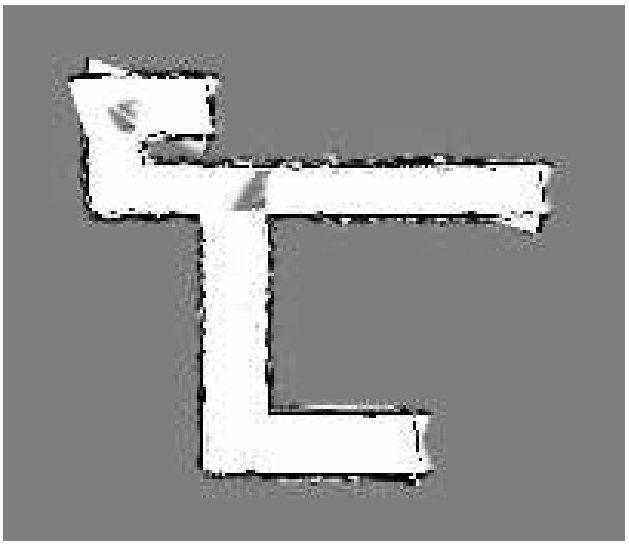

Fig. 7. Map Created with Grid Configuration 1 with 3 Robots Mapping

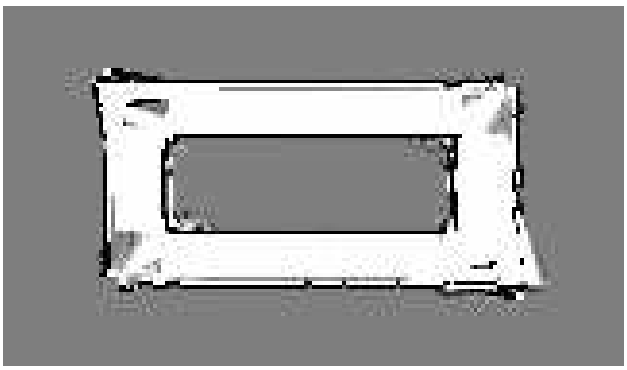

Fig. 8. Map Created with Grid Configuration 1 with a Single Robot in Simulation

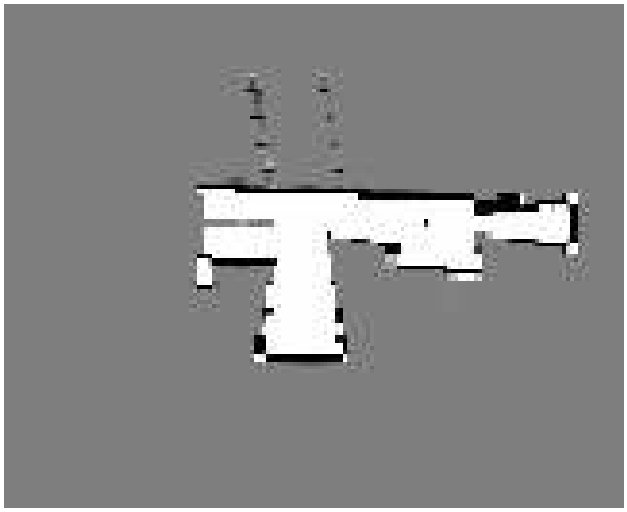

Fig. 9. Map Created with Grid Configuration 1 with Nomad 200

Fig. 10. Map Created with Grid Configuration 2 with 1 robot mapping in Simulation

Grid configuration 2 did not present the same problem as grid configuration 1. The robot did not pause when writing to the database and accurate maps were created. This performance increase is due to the efficiency is due to grid configuration 2's efficient storage allocation. However, if the number of sensor readings per update is increased significantly, a noticable pause becomes apparent. It is also worth noting that as the efficiency of space allocation is increased from grid configuration 1 to grid configuration 2, the complexity of determining whether space has already been allocated significantly increases.

VI. Conclusions and Future Work

We have shown a technique for creating dynamically expanding occupancy grids to solve the navigation problem without any pre-specifications for the size of the map and construct the map in an inherently dynamic fashion. We have also illustrated that our database storage scheme is both robust and is a natural solution to occupancy grid storage. We have demonstrated that efficient storage and

bandwidth can be achieved through dynamic space allocation with a centralized storage location using adaptive grid configurations. Grid configuration 2 is also natuarally adaptive to heterogeneous multi-agent systems.

Our method performs comparably to existing methods, but introduces a more effecient and stable system. Storage efficiency is improved through the use of dynamic space allocation and a centralized, global, storage unit. The stability of the system lends itself to the use of TCP communication, which differs from typical broadcast systems which use UDP. The proposed method is computatinally sound and has a simplistic and well studied representation.

One obvious expansion of this technique would include producing extremely large scale maps (on the size of 10^8 ft^2) with more agents. With the scalability of database management systems, this should be a trivial task. Also, the application of heterogeneous robot systems and sensing systems should be tested with grid configuration 2. Given the varying degrees of accuracy in various domains for particular sensors, this method should adequately handle any distance finding sensors.

Another possible expansion is to setup a hierarchical system of centralized storage locations. Agents could then update the centralized storage location within the nearest range which would in turn up date the next upper layer unit continuing until the uppermost unit is reached.

This method is applicable to numerous types of applications. Any application that requires the mapping of large areas by many unmanned robots, for safety or efficiency reasons, could benefit from this technique. Dangerous and possibly unknown environments could be quickly mapped or scouted by teams of robots with little to no human intervention or contact.

REFERENCES

[1] H.P. Moravec and Alberto Elfes. High resolution maps from wide angle sonar. *IEEE International Conference On Robotics and Automation*, pages 116–121, 1985.

[2] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[3] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer Magazine, special issue on Autonomous Intelligent Machines*, 22(6):46–58, 1989.

[4] Bharani Kumar Ellore. Dynamically expanding occupancy grids. Master's thesis, Texas Tech University, November 2002.

[5] Jean-Claude Latombe Christopher M. Clark, Stephen M. Rock. Motion planning for multiple mobile robots using dynamic networks. In *IEEE International Conference On Robotics and Automation*, 2003.

[6] Martin C. Martin and Hans Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

[7] Andrew Howard and Les Kitchen. Sonar mapping for mobile robots. 1996.

[8] Wolfram Burgard, Dieter Fox, Hauke Jans, Christian Matenar, and Sebastian Thrun. Sonar-based mapping with mobile robots using EM. In *Proc. 16th International Conf. on Machine Learning*, pages 67–76. Morgan Kaufmann, San Francisco, CA, 1999.

[9] Kurt Konolige. Improved occupancy grids for map building. In *Autonomous Robots*, volume 4, 1997.

[10] Tucker Balch. Grid-based navigation for mobile robots.

[11] D.J. Cook G.M. Youngblood, L.B. Holder. A framework for autonomous mobile robot exploration and map learning through the use of place-centric occupancy grids. *ICML Workshop on Machine Learning of Spatial Knowledge*, 2000.

[12] Y. Koren and J. Borenstein. Potential fi eld methods and their inherent limitations for mobile robot navigation. *Proceedings of the IEEE Int. Conference on Robotics and Automation*, pages 1398–1404, 1991.

[13] A. Howard and L. Kitchen. Generating sonar maps in highly specular environments. *Proceedings of the Fourth International Conference on Control, Automation, Robotics, and Vision*, pages 1870–1874, 1996.

[14] B. Yamauchi, A. Schultz, and W. Adams. Integrating exploration and localization for mobile robots. *Autonomous Robots, Special Issue on Learning in Autonomous Robots*.

[15] Clark F Olson. Subpixel localization and uncertainity estimation using occupancy grids. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, 1999.

[16] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artifi cial Intelligence*, 128(1-2):99–141, 2001.

[17] M. Ribo and A. Pinz. A comparison of three uncertainty calculi for building sonar-based occupancy grids. *International Joint Robotics and Autonomous Systems*, 35:201–209, 2001.

[18] Hans Moravec and Mike Blackwell. Learning sensor models for evidence grids. *Robotics Institute Research Review*, September 1992.

[19] Nomadic Technologies. Nomad 200 users's manual. 1997.