Proceedings of the 1998 IEEE
International Conference on Robotics & Automation
Leuven, Belgium • May 1998

A Graph-Based Exploration Strategy of Indoor Environments by an Autonomous Mobile Robot

Jane Yung-jen Hsu* Liang-Sheng Hwang
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan 106, R.O.C.

Abstract

This paper presents a provably complete *strategy for indoor environment exploration by an autonomous mobile robot. Without prior knowledge about the environment, the strategy guarantees the construction of a grid-based map of the entire reachable area within a bounded region. Multiple map representations are utilized including a topological grid map for guiding the exploration process, a modified occupancy grid for fusing data from multiple range sensors, and a hierarchy of grids for real-time navigation. Experiments using a Nomad 200™ robot have shown accurate map construction while navigating at a steady speed of 0.2m/sec.*

1 Introduction

There are many potential applications of mobile robots in our everyday life. Imagine a floor-cleaning robot that plans a priori the *optimal* path to clean an area using a map. To ensure successful completion of the task, access by other agents to the area has to be restricted. Moreover, the task may fail if the layout of furniture has changed. For applications like household vacuuming, demanding an accurate map is simply impractical.

A mobile robot operating in the real world needs to deal with such incomplete and uncertain information about its environment. Although sensor-based navigation techniques enable a robot to negotiate its way through unknown obstacles without a map, for more effective navigation and task execution, it is useful for it to construct and update its world model dynamically [2]. An autonomous mobile robot should 1) have an *exploration strategy* to guide its search, 2) construct a world model by fusing data collected through multiple sensors, and 3) utilize the dynamically updated map for real-time navigation and task achievements.

*This research was supported in part by the National Science Council of ROC under grant NSC 86-2212-E-002-024.

This paper presents a graph-based exploration strategy that is provably complete with respect to any bounded region, e.g. indoor environments. The exploration results in a grid-based map constructed on-line by fusing range data from multiple sensors while the robot navigates in an initially unknown indoor environment. The resulting map consists only of static features as desired. In Section 2, we begin by briefly outlining the proposed graph-based search algorithm, followed by detailed descriptions and the completeness proof in Section 3. Section 4 presents a simplified certainty grid method for fusing data from multiple range sensors according to the sensor characteristics. The proposed method achieves real-time updating while preserving enough accuracy for global planning and navigation within the environment. Experiments on a Nomad 200™ mobile robot with sonar, infrared, and laser range sensors are presented in Section 5. The results showed that the system is efficient, effective and robust in a dynamic environment.

2 Graph-Based Exploration Strategy

Terrain exploration has been an important research problem for autonomous mobile robots [10, 9, 4, 5]. In this work, a *graph traversal algorithm* is proposed to guide a robot in exploring its environment so that every reachable area will be visited. The algorithm also minimizes the overhead in moving from one exploration point to another.

An unknown bounded region can be decomposed into a two-dimensional tessellation. The size of each cell in the tessellation is decided by the radius of the robot's effective sensor range. Each cell is mapped into a vertex in the corresponding graph, whose connectivity is defined by the special adjacency relation among cells. Figure 1 shows a sample mapping into the graph representation. Such a topological graph records the status of each cell as one of *unknown*, *visited*, *passed*, or *inaccessible*.

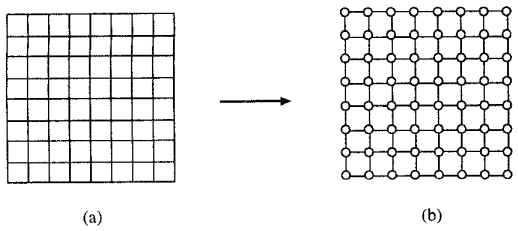

Figure 1: The topological grid of a bounded region

3 Algorithms and Completeness

The exploration strategy is summarized as follows.

1. Initialize every cell to *unknown*.
2. $c \leftarrow$ the cell where exploration starts.
3. **loop until** every cell is *visited* or *inaccessible*;

   loop until obstacle detected at c;
   Move straight to next cell c';
   Mark c as *passed*;
   $c \leftarrow c'$.
   end
   if c is *not visited*,
   then Circumnavigate obstacle boundary;
   Update geometric world model;
   Mark cells travelled as *visited*;
   Fill cells inside obstacle as *inaccessible*.
   $c' \leftarrow$ *Best-Next-Cell*(c)
   Navigate to c'.
   Mark cells on the path as *passed*.
   $c \leftarrow c'$.
   end

Figure 2: The Graph Traversal Algorithm

3.1 The Filling Algorithm

The filling algorithm marks the *inner* cells of an obstacle as *inaccessible*. Due to poor resolution of grids, standard filling algorithms such as the *even-odd* needs to be modified. Cells on the boundary of a grid-type polygon are viewed as vertices. A scan line may intersect with a polygon at a continuous segment of boundary cells, which are called an *intersection unit* (IU) (e.g. Figure 3 left). Each IU is treated as a single intersection point. If the cell immediately before and after an IU straddle the scan line (as shown in Figure 3 right), it is a *breaking* IU.

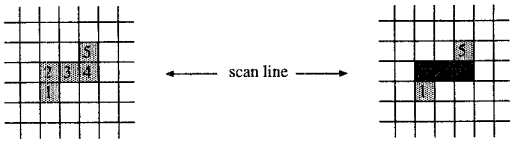

Figure 3: A sample IU consisting of cells 2, 3, and 4. Cells 1 and 5 straddle the scan line.

Figure 4(a) illustrates the sequence of a typical trajectory generated by our robot. Figure 4(b) shows the corresponding boundary chain. Note that the subsequence of cell 22 is cell 26; the subsequence of cell 31 is cell 33. There are three IUs (IU1[30/32,29,28], IU2[2/24], and IU3[48,47]) on the indicated scan line. IU1 will be misjudged as a breaking IU, which leads to an incorrect fill! The misjudgment of IU1 is due to the "jumping connectivity" of cells ($31 \rightarrow 33$).

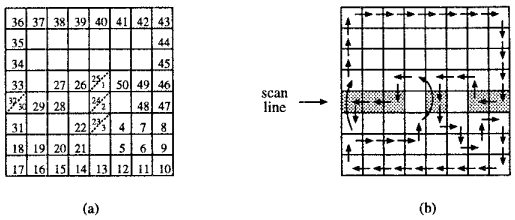

Figure 4: (a) The sequence of cells passed by the robot, and (b) its corresponding boundary chain.

The problem can be solved by relaxing the constraint that only one visit is permitted for each cell in the polygon boundary. The algorithm for computing the breaking IUs are summaried below.

Requirement:

A vertex chain V of polygon boundary without jumping connectivities.

A tessellation buffer *tess* initialized with 0's.

Pass1: for each vertex v_i in V, do
$\Delta Y \leftarrow \text{CoorY}(\text{succ}(v_i)) - \text{CoorY}(\text{pre}(v_i))$
$tess[\text{CoorY}(v_i)][\text{CoorX}(v_i)]$
$\leftarrow tess[\text{CoorY}(v_i)][\text{CoorX}(v_i)] + \Delta Y$

Pass2:
for each horizontal scan line l, do
for each *intersection unit* P on l do
$t \leftarrow \sum_{v_i \in P} tess[\text{CoorY}(v_i)][\text{CoorX}(v_i)]$
if $t \neq 0$, P is a breaking IU

The function succ (pre) takes a vertex as argument and returns the succeeding (preceding) element in the

chain V. The function CoorX (CoorY) takes a vertex as argument and returns its x (y) coordinate. Figure 5 (b) shows the results given the boundary in Figure 5 (a).

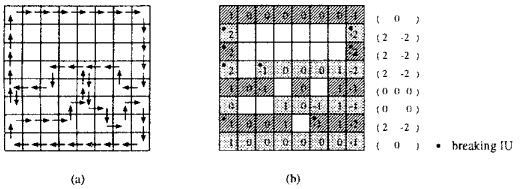

Figure 5: Results of intersection computation

3.2 Selecting the Best Next Node

Each cell in the tessellation T can be in one of the following four states:

Unknown (U): the cell is unknown to the robot; all cells are in this state initially.

Boundary (B): the cell has been visited by the robot while it is in the circumnavigation mode.

Pass (P): the cell has been visited by the robot while it is *not* in the circumnavigation mode.

Forbidden (F): the cell is marked as forbidden in the filling process.

Our goal is to select the "best" cell in state U as the next exploration target. Not all cells are reachable from the current position of the robot and there may be unobserved obstacles. However, we can always navigate to a cell in state U that is next to some visited cells. We first identify as target any cell in state P or B having unvisited neighbors. The robot will then proceed to explore one of its unknown neighbors.

The simple *shortest distance first* heuristic is used in choosing the best next node. The idea is similar to numerical potential field [1]. Imagine the closed environment as a pond and the forbidden cells as "islands" in it. Ripples propagate outward from the position of the robot. Assuming there are no reflected waves, the first candidate cell encountered by the waves is the target. More precisely,

$$u_{k+1}^r = \begin{cases} k+1 & \text{if } u_k^r = 0 \text{ and } \exists x \in \aleph_4(u_k^r) \text{ s.t. } u_k^x > 0 \\ u_k^r & \text{otherwise} \end{cases} \quad (1)$$

where $k = 1, 2, 3 \ldots$. The initial values are defined as:

$$u_1^x = \begin{cases} 1 & \text{if the robot is currently at cell } x \\ -1 & \text{if } x \text{ is a forbidden cell} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

u_{k+1}^r denotes the status of a cell r with respect to the ripples at time step $k+1$: 0 indicates it has not been reached by any ripple; i indicates it was first reached by the ith ripple; -1 indicates a forbidden cell. $\aleph_4$ is a function that takes a cell as argument and returns the set of its four neighbors. Figure 6 shows the values of such a computation.

9	8	7	6	5	4	5	6
10	9	8		4	3	4	5
9	8				2	3	4
8	7	6			1	2	3
7	6	5	4	3	2	3	4
8	7		5	4	3	4	5
9				5	4	5	6
10	9	8	7	6	5	6	7

(a)

forbidden cell

Figure 6: The numerical ripple.

3.3 Completeness of the Strategy

The section presents some formal results for the proposed approach.

Lemma 1 *The procedure described in Section 3.2 always selects an unknown cell if one exists.*

Lemma 2 *The proposed graph-based terrain acquisition algorithm terminates.*

Theorem 1 (Completeness) *When the program terminates, all reachable cells are visited by the robot.*

Proof By Lemma 2, when the algorithm terminates, all cells are known to the robot, either visited or forbidden. Since the filling algorithm correctly fill the inside and outside of a polygon, all reachable cells will be correctly classified into state B or F.

We can also show that the total distance traversed by the robot has an upper bound of $O(N^2)$ and a lower bound of $O(N)$, where N is the total number of reachable cells in the environment.

4 World Modeling

Occupancy grids [6, 7] or *certainty grids* [11] provide a good way to fuse sensor data over time. Using the Bayes' rule, one can compute the improved probabilistic estimates by integrating a new observation with the existing model. Several modifications have been proposed to allow grid values based on fuzzy [12] or evidential [13] theories. However, updating the grid values is very computationally intensive. Given an area of $n * n$ cells, it takes $O(2^{n^2})$ amount of computation to update the probability of whether a single cell C_i is occupied when a new sensor reading is obtained. If the values are calculated off-line for reuse, a table of size $O(2^{n^2})$ is needed. As a result, occupancy grids are feasible only within a local region [14] since the map may not reflect in time dynamic changes in the environment. On the other hand, computing the *vector field histograms* [3] is simple, but the resulting model is only good for local obstacle avoidance rather than global path planning.

This research proposed a simplified computation rule for updating a *grid-based geometric map* that produces reasonably accurate results for navigation. The proposed approach can be used to integrate data from multiple sensors according to the different sensor characteristics. The size of the cell is determined by the highest sensor resolution as well as the minimal performance requirements. The basic idea is to associate each cell C_i with a real number, called *certainty value*, which indicates how likely the cell is being occupied by an obstacle. Let $V(C_i)$ denote the measure of confidence that an obstacle exists within cell C_i. A higher certainty value indicates a stronger evidence for believing the cell to be occupied. The grid-updating procedure can be summarized below, where ϵ is a small number as the default certainty value.

1. Initialize each cell $V(C_i) = \epsilon$.
2. **loop until** exploration is completed.
   - Obtain the current sensor readings r.
   - Interpret r using *occupancy probability profile.*
   - Fuse multiple sensor readings by *sensor weight functions.*
   - *Update certainty value* for each cell.
   - **end**

Figure 7: The cell updating cycle

4.1 Occupancy probability profile

The occupancy probability profile specifies the area of interest, i.e. the set of cells relevant to the current reading. Given a sensor reading r for an *ideal* sensor, a possible profile is defined as follows:

$$P(x) = \begin{cases} 0 & \text{if } x < r \\ 1 & \text{if } x = r \\ \frac{1}{2} & \text{if } x > r \end{cases} \quad (3)$$

where x is any cell along the direction of the detected obstacle. That is, the probability is 1 for some object to be within the cell boundary corresponding to the given sensor reading, and it is 0 for all intermediate cells between the robot and the detected obstacle. The cells behind the detected obstacle are not observable from the vintage point of the robot and therefore are equally probable to be occupied or empty, i.e. a probability of $\frac{1}{2}$.

In general, the scanned obstacle may lie anywhere within a cone-shaped neighborhood along the acoustic axis. Considering the trade-offs between computation and accuracy, a combined approach was employed in our implementation: the "line model" is adopted when something is detected by the sonar; otherwise, use the "area model". The intuition was that the area model is significantly harder to compute in the former case, yet its benefit is uncertain.

Unfortunately, indoor experiments on real robots have shown that echoing and deflection of the sonar sensors present a serious problem under the area model: map corruption occurs too frequently, especially in the regions near the robot's trajectory. Since sonars use the *time-of-flight* of the reflected signal to compute the distance to the sensed obstacle, the intensity of reflecting waves depends on the incident angle between the acoustic axis of the sonar and the normal of the obstacle. Inaccurate measurements are an inherent problem with the ultrasonic sensors.

To overcome such problems, the idea of a *certainty momentum function* is introduced. The momentum is defined as a function of the *certainty values* rather than of *distance*. Since the certainty value of any cell has to be computed incrementally, a high certainty value can be accumulated only if the cell has been estimated to be occupied for a large number of times. It makes sense to ignore an occasional "wrong" reading. A typical momentum function defines cells with higher values to have a higher probability to stay occupied by an obstacle. When nothing is detected by the sonar, cells with certainty values over a threshold will remain intact to avoid map corruption. With continuous and

rapid sampling, the approach generates reasonable results in our experiments. More details can be found in [8].

4.2 Sensor Weight Functions

The objective of fusing data from multiple sensors is to produce a robust, consistent, and fault-tolerant description of the environment. None of the range sensors is perfect. Each type of sensors has its own characteristics, and multiple sensors can be combined to compensate for one another. For example. the laser range finder produces accurate readings, but its effective range is smaller than the sonar; infrared data varies with lighting conditions; sonar readings are affected by surface materials.

In this framework, two weights are associated with each sensor: *sensor type*, W_{S_i}, and *distance*, W_{dis}. The former is defined in terms of the relative accuracy of a particular type of sensor. Given a sensor s, its accuracy $A(s)$ is defined as:

$$A(s) = \sum_{d \in \mathcal{R}} \frac{1 - \frac{|r-d|}{d}}{\#d} \tag{4}$$

where r is the range reading, d is the actual distance, $\mathcal{R}$ is the effective range, and $\#d$ is the number of measurements taken. Three types of range sensors including laser, sonar, and infrared are used in our experiments. Their accuracies are ordered as $A(ls) > A(ir) > A(sn)$. Let $\mathcal{A}$ be the sum of $A(ls) + A(sn) + A(ir)$. The *sensor type weights* are defined as follows.

$$W_s = \frac{A(s)}{\mathcal{A}} \tag{5}$$

The latter, *distance weight*, represents the relative accuracy of a given sensor over its effective distance. Let W^s_{dis} denote the distance weight function of sensor s. Figure 8 shows the different W_{dis} functions of laser, sonar, and infrared. To model the decayed density of reflected signal with the increase in distance to the robot, sigmoid functions were used for sonar and infrared. On the other hand, the laser is modeled by the normal distribution function to capture its utility within a limited range.

The total weight for a specific sensor s with reading r is calculated by combining the two weights.

$$\mathcal{W}^s(r) = W_s * W^s_{dis}(r) \tag{6}$$

Such sensor characteristics are often available from its specifications as well as through empirical tests.

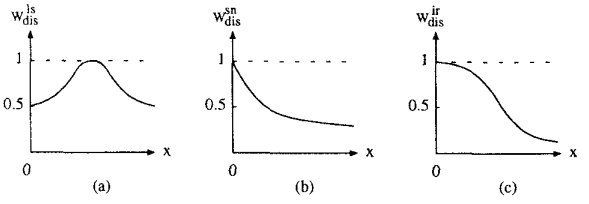

Figure 8: Weight functions of different sensors.

The weight functions can be computed off-line and stored in look-up tables to speed up subsequent map updating.

4.3 Certainty Value Updates

We are now ready to present the formula for updating certainty values during each sensing cycle. To compose the sensory information incrementally, one needs to combine the cumulative value with the new observation. The *sequential update momentum function* $m(x)$ is used to define the relative weight of the two. Some typical momentum functions are shown in Figure 9. For example, Figure 9(a) shows that the relative weight decreases smoothly with the decrease of distance to the robot. This comes from the intuitively appealing result that the areas around the robot has a higher tendency of being updated than those farther away.

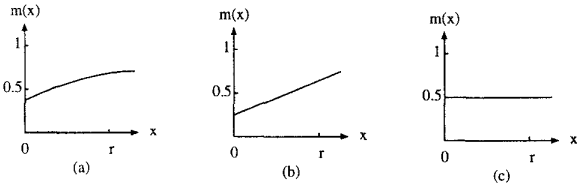

Figure 9: Sequential update momentum functions

For each cell x, a gain (loss) of credibility is added to its certainty value $C(x)$ if it is judged to be occupied (empty) in the current cycle. Assume that the range sensor s returns a reading of r at stage k. The certainty value $C_k(x)$ of cell x at stage k is:

$$C_k(x) = \begin{cases} \alpha * C_{k-1}(x) + \beta * gain & \text{if } 0 \le x \le r \\ C_{k-1}(x) & \text{otherwise} \end{cases} \tag{7}$$

where *gain* is a constant value, $\alpha = m(x)$ is the momentum function, and the update ratio β is:

$$\beta = (1 - \alpha) * P(x) * \mathcal{W}^s \tag{8}$$

Due to frequent map updating, the world model can dynamically reflect changes, such as moving obstacles, in the environment. As a result, the path planner can replan a better path whenever substantial changes have occurred in the map.

5 Experiments

To verify the feasibility of the proposed method, experiments have been performed both in a simulated environment and on a mobile robot. The Nomad 200™ is an integrated mobile robot system with four sensory modules including tactile, infrared, ultrasonic, and structured light vision system. It has an onboard computer for sensor and motor control as well as for host computer communication. The mobile base keeps track of its position and orientation over time. The tactile system consists of 20 independent pressure sensitive switches arranged in two rings, which detect contacts with an object. The 16-channel, reflective intensity based infrared ranging system provides information up to 30 inches, under proper conditions. The 16-channel, time of flight based ultrasonic ranging system provides information from 17 inches to 256 inches. The two-dimensional, triangulation based laser ranging system has an operating range of 12 to 120 inches. The initial certainty values were $V(C_i) = 5$ in our experiments.

Figure 11 shows the results from experiments in a simulated environment with static obstacles (Figure 10. The resulting map has IOP=2.5744 in. and IOA=99.06%.

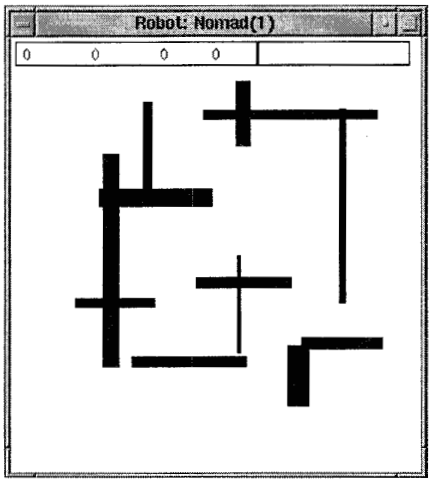

Figure 10: A simulated world with static obstacles

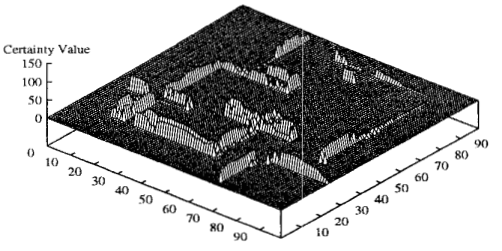

Figure 11: Learned map for the simulated world

To demonstrate the effects of dynamically moving objects within the environment, a series of experiments were performed in a real-world environment Two irregularly shaped thin cardboards were laid out in the middle of the room with a person moving about within the space throughout the experiments. Figures 12 and 13 shows the results. In particular, the cardboards were positively identified, while the moving person didn't show up on the map. Using the map, the robot was able to navigate freely within the space without bumping into any obstacles.

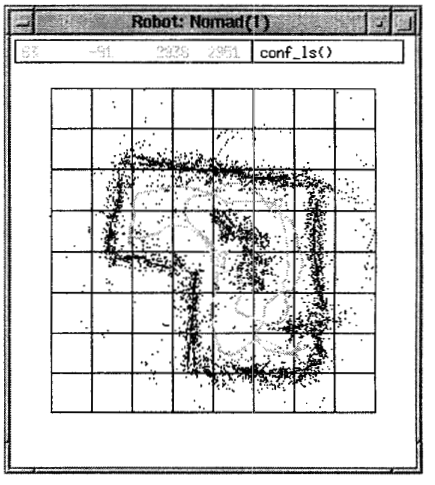

Figure 12: Sensor traces in a real environment

6 Conclusions

This paper presented a graph-based exploration strategy for a mobile robot in indoor environments.

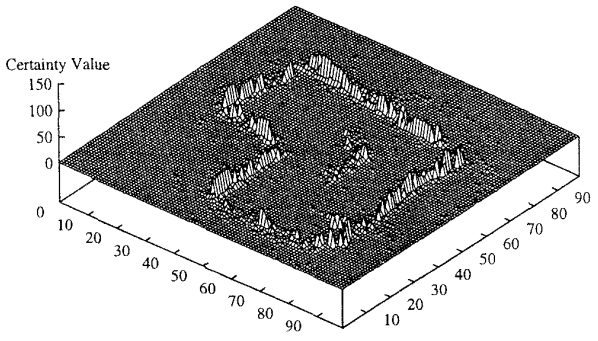

Figure 13: Results in an environment with dynamic obstacles (IOP=9.0056 in., IOA=95.8679%)

Our approach utilizes different levels of maps for exploration control, world modeling, and real-time navigation. In particular, the proposed strategy guarantees complete exploration of any bounded enclosed area. The overall performance has been shown to be both robust and efficient. The current system uses standard sensor-based postion estimation techniques [2] to correct its cumulative odometry errors. While the resulting map is adequate for navigation of service robots for house cleaning, better localization techniques in unknown environments will be necessary for higher precision tasks.

References

[1] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241, April 1992.

[2] J. Borenstein, H. R. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques.* A.K. Peters, 1996.

[3] J. Borenstein and Y. Koren. The vector field histogram—fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991.

[4] Z. Chen and C.-M. Huang. Terrain exploration of a sensor-based robot moving among unknown obstacles of polygonal shape. *Robotica*, 12:33–44, 1994.

[5] T. Edlinger and E. von Puttkamer. Exploration of an indoor-environment by an autonomous mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1278–1284, September 1994.

[6] A. Elfes. Sensor-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3:249–265, June 1987.

[7] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference on Certainty in AI*, July 1990.

[8] L.S. Hwang. Automatic map building for a mobile robot. Ms thesis, National Taiwan University, June 1995.

[9] J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox. Dynamic map building for an automous mobile robot. *The International Journal of Robotics Research*, 11(4):286–298, August 1992.

[10] V. Lumelsky, S. Mukhopadhyay, and K. Sun. Sensor-based terrain acquisition: The "sightseer strategy". In *Proceedings of the 28th Conference on Decision and Control*, pages 1157–1161, Tampa, Florida, December 1989.

[11] H. P. Moravec. Sensor-fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[12] G. Oriolo, M. Vendittelli, and G. Ulivi. On-line map building and navigation for autonomous mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2900–2906, 1995.

[13] D. Pagac, E. M. Nebot, and H. Durrant-Whyte. An evidential approach to probabilistic map-building. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 745–750, April 1996.

[14] F. Wallner, R. Graf, and R. Dillman. Real-time map refinement by fusing sonar and active stereo-vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2968–2973, 1995.