

# An Integrated Mobile Robot Path (Re)Planner and Localizer for Personal Robots\*

Jinsuck Kim Nancy M. Amato  
Department of Computer Science  
Texas A&M University  
College Station, TX 77843  
{jinsuckk, amato}@cs.tamu.edu

Sooyong Lee  
Department of Mechanical Engineering  
Texas A&M University  
College Station, TX 77843  
slee@mengr.tamu.edu

## Abstract

*Personal robotics applications require autonomous mobile robot navigation methods that are safe, robust, and inexpensive. Most of the previous techniques proposed do not meet these competing goals. In this paper, we describe a method for navigation in a known indoor environment, such as a home or office, that requires only inexpensive range sensors. Our framework includes a high-level planner which integrates and coordinates path planning and localization modules with the aid of a module for computing regions which are expected, with high probability, to contain the robot at any given time. The localization method is based on simple geometric properties of the environment which are computed during a preprocessing stage. The roadmap-based path planner enables one to select routes, and sub-goals along those routes, that will facilitate localization and other optimization criteria. In addition, our framework enables one to quickly plan new routes, dynamically, based on the current position as computed by intermediate localization operations. We present simulation and hardware experimental results that illustrate the practicality and potential of our approach.*

## 1 Introduction

There is an increasing number of potential applications for autonomous mobile robots in indoor environments. Personal robotics applications must be safe, robust, and inexpensive.

Unlike manipulator robotics in many manufacturing applications, mobile robotics requires a global understanding of the environment and the ability to dynamically plan in it [8]. There are some fundamental issues that must be addressed to achieve autonomous mobile robot navigation. First, a *path planner* is needed to select a route for the robot to follow (see, e.g., [12]).

---

\*This research supported in part by NSF CAREER Award CCR-9624315 (with REU Supplement), NSF Grants IIS-9619850 (with REU Supplement), ACI-9872126, EIA-9975018, EIA-9805823, and EIA-9810937, and by the Texas Higher Education Coordinating Board under grant ARP-036327-017.

Second, since unavoidable odometer errors, e.g., due to wheel slippage, render it impossible for any mobile robot to precisely follow a planned trajectory, *localization* techniques are needed to precisely determine the robot's position and orientation (see, e.g., [3, 4, 6, 5, 9, 15, 17, 18]).

In this paper, we describe the design and implementation of a framework for autonomous mobile robot navigation in a known indoor environment that requires only inexpensive range sensors. Our approach:

- integrates and coordinates the path planning and localization modules with the aid of a module for computing *uncertainty ellipses*.
- selects routes, and sub-goals along those routes, that facilitate localization and optimization of other criteria such as travel time or safety.
- dynamically re-plans routes quickly and easily from the current configuration

Our framework is based on a roadmap-based path planner [12] and our recently proposed localization strategy [13], which requires only inexpensive range sensors. An advantage of roadmap planning methods is that the presence of multiple potential paths in the roadmap provides support for on-line re-planning of the path, and enables one to select routes which optimize various criteria. Our localizer assumes that a description of the environment is available *a priori*, and that the portion of the workspace relevant to navigation can be modeled with polygonal obstacles.

## 2 Global Navigation

The navigator takes as input **start** and **goal** configurations of the mobile robot, and the known environment information. It calls on a *path planner* to plan a path, and then determines a trajectory. Due to imprecise control and odometer error, the mobile robot cannot precisely follow the planned trajectory. Thus, several iterations of this process might be required before

the robot attains the **goal** configuration. Before each iteration, a *localization* will be performed to precisely determine the robot’s current configuration, which will be the new **start** configuration by the path planner.

In many cases the path output by the path planner will be modified to facilitate the subsequent localization step and also to ensure the robot follows a valid path (e.g., to ensure that no collisions will occur due to imprecise control or odometer error). In particular, some upper bound on the positioning error the robot will accumulate as it travels will be known based on the specifications of the mobile robot. These bounds will be used to determine a region that is expected to contain the robot as it travels – these regions are often called *uncertainty regions* (ellipses in our case). To ensure that no collision will occur, the navigator must plan a trajectory in which the uncertainty region does not intersect any obstacles. In addition, if the localization algorithm used has any special requirements (as does the method we employ), then the navigator must also be sure that the uncertainty regions on the trajectory could not place the robot in a situation where localization is not possible.

Thus, moving the mobile robot from a given **start** to a **goal** configuration is an iterative process that is governed by a high-level navigator which synthesizes the current trajectory to be given to the robot from information provided by path planning, localization, and uncertainty/error calculation modules. A pseudo-code description of the process is shown in Figure 1.

```

NAVIGATOR(start, goal)
1. while goal is not reached {
2.   find path from start to goal
3.   compute uncertainty regions along path
4.   determine subgoal ('safe' prefix of path)
5.   compute trajectory from start to subgoal
6.   drive robot to subgoal and stop
7.   robot senses environment
8.   localize robot using sensor input
9.   set start to current configuration
10. }

```

Figure 1: Pseudo-code for Global Navigator

The main intelligence required by the navigator is in Step 4, the determination of an appropriate **subgoal** along the path from the current position to the desired final **goal**. This requires an understanding of the capabilities and requirements of the localization method. For example, if there are certain situations in which localization would not be possible, then the global navigator must ensure that they do not occur.

In addition, the navigator is responsible for performing any global optimization of the selected route. For example, if sensing and localization is a time-consuming operation, and speed is a concern, then the

navigator must consider both the length of the path and the number of localization operations required when selecting a route. In this case, it will aid the navigator’s task if the path planning module is capable of providing multiple paths from the **start** to the **goal**. For this reason, we favor *roadmap* path planning algorithms [12], which encode multiple representative feasible paths.

### 3 Component Algorithms

In this section, we describe the path planning, localization, and uncertainty/error computation modules used in our system. We assume that the portion of the workspace relevant for navigation can be considered to be planar, and that obstacles can be modeled by simple polygons. Since our goal is to design a navigation system for inexpensive robots, we have selected a localization method that requires only range (distance) data, which can be obtained by low cost sensors.

#### 3.1 Roadmap Path Planning

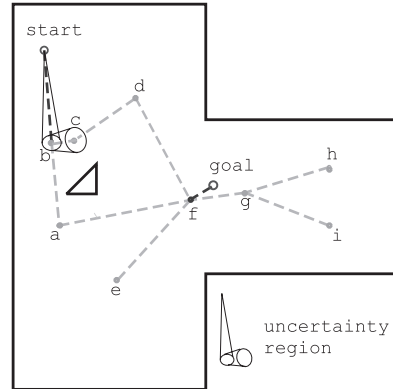


Figure 2: Roadmap and Uncertainty Region

Roadmap motion planning methods [12] construct, usually during preprocessing, a graph of representative feasible paths which can later be used to answer many, varied motion planning queries. In theory, we can use any roadmap. One attractive option is to use roadmaps built on the medial axis of the workspace or C-space [7, 15, 19], which tend to maximize the robot’s clearance from obstacles. While computing the medial axis in high-dimensional C-spaces can be expensive, many important applications for mobile robots can be framed in a planar, polygon environment. In this case, computation of the medial axis is not prohibitive.

Assuming a roadmap is available, queries are answered by connecting the **start** and **goal** configurations to the same connected component of the roadmap

using some ‘local planner,’ and then extracting a path connecting them using graph search methods. An example is shown in Figure 2, where the **start** and **goal** configurations are connected to the roadmap nodes  $b$  and  $f$ , respectively, and the path extracted is composed of the nodes  $b, c, d$ , and  $f$ .

### 3.2 Uncertainty Regions

The count of wheel revolutions provides positional information relative to the start location. This kind of measurement is called “dead-reckoning.” Since odometer readings are subject to unavoidable errors, much work has been done on estimating the position uncertainty of a dead-reckoning robot [11]. Typically with this approach, each computed robot position is surrounded by a characteristic *error ellipse* which indicates a region of uncertainty for the robot’s actual position [1], i.e., the ellipse denotes a region that will contain the robot with high probability.

For a straight translational path, the ellipse at the end of the path can be approximated using two parameters corresponding to longitudinal and lateral deviation and can be visualized as the minor and major axes of an uncertainty ellipse. The uncertainty region of a translational path is obtained by taking the intersection of the ellipses along the path. (see Figure 2).

### 3.3 Localization

Many different localization methods have been proposed in the literature (see, e.g., [3, 4, 5, 6, 9, 15, 17, 18]). We have selected a method we developed that provides fast localization using only range sensor data (i.e., distance measurements) [13]. During preprocessing, the workspace is partitioned into sectors using simple visibility computations, and a small identifying label is computed for each sector. The localizer analyzes the range sensor readings and extracts characteristic points, which are compared with the pre-computed sector labels to localize the robot, first to a sector, and then to a particular configuration within that sector. This two step process is computationally very simple, and allows precise localization at any place in the workspace without any landmarks.

**Visibility Sectors.** Two points  $p_1$  and  $p_2$  are *visible* if  $\overline{p_1 p_2}$  does not properly intersect any obstacle. The *point visibility polygon*  $V(p)$  is the set of all points visible from  $p$  [16]. A *visibility sector*  $s_i$  is a maximal region such that every point  $p \in s_i$  is visible from the same set of environment vertices. Thus, the visibility sectors of the environment are the faces in the planar subdivision that is obtained by overlaying the point visibility polygons for all environment vertices (see Figure 3).

Our visibility sectors are the same constructs called

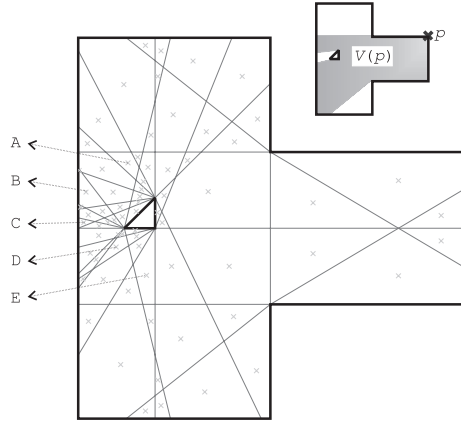


Figure 3: A point visibility polygon and sectors

visibility cells in [10], where it is shown that given a simple polygon  $P$ , the complexity of the decomposition is bounded by  $O(n^2 r)$ , where  $n$  is the number of vertices of  $P$  and  $r$  is the number of reflex vertices.

**Sector and Scan Labels.** We compute, during preprocessing, a *label* for each sector that includes a component for each vertex and edge of the environment visible from that sector. Similar labels will be constructed from the scan data sensed by the mobile robot. The component for each feature is classified as one of four types of *characteristic points*. The *local maxima* (M), *discontinuity points* (D), *local minima* (m) and *connection point* (c).

A sector’s label is a string of mark characters (M, m, D and c), one for each characteristic point, which appear in the order the characteristic points are seen in a counter-clockwise scan with an initial orientation of 0 (i.e., eastern point first). The scan label is constructed by processing the range sensor readings, also in counter-clockwise order, as described in the pseudo-code shown in Figure 4, where  $R = \{r_0, r_1, \dots, r_{N-1}\}$  is the set of  $N$  distance readings sorted in angular order, all index arithmetic is done modulo  $N$ , and  $\circ$  denotes string concatenation.

The sector labels and the algorithm for constructing the scan label are minor modifications of the those presented in [13]. In particular, they have been modified so that sector boundaries perpendicular to obstacles, such as the segment labeled **a** in Figure 5, are not needed. If such boundaries were kept, we would have adjacent sectors whose labels differed only by the inclusion or exclusion of local minimum characteristic points. Such points represent internal points on edges that are not useful for precise localization since they do not have fixed global coordinates. Therefore, eliminating such points from our labels enables us to have fewer sectors without increasing the complexity of localization. Thus, for example, the algorithm has been

```

CONSTRUCT SCAN LABEL( $R$ )
1.  $\ell_R := \emptyset$ 
2. for  $i := 0$  to  $N$ 
3.   if  $(r_i - r_{i+1} > \varepsilon_T)$  then
4.     if  $(r_{i+1} < r_{i+2})$  then  $\ell_R := \ell_R \circ cD$ 
5.     else  $\ell_R := \ell_R \circ cD$ 
6.   elseif  $(r_{i+1} - r_i > \varepsilon_T)$  then
7.     if  $(r_i < r_{i-1})$  then  $\ell_R := \ell_R \circ mDc$ 
8.     else  $\ell_R := \ell_R \circ Dc$ 
9.   elseif  $(r_{i-1} > r_i$  and  $r_i < r_{i+1}$  and
       $r_{i-1} - r_i < \varepsilon_T)$  then
       $\ell_R := \ell_R \circ m$ 
10.  elseif  $(r_{i-1} < r_i$  and  $r_i > r_{i+1}$  and
       $r_i - r_{i-1} < \varepsilon_T)$  then
       $\ell_R := \ell_R \circ M$ 
11.  endif
12. endfor

```

Figure 4: Pseudo-code for scan label construction.

modified so that the same scan labels will be computed from scans collected at A or B in Figure 5.

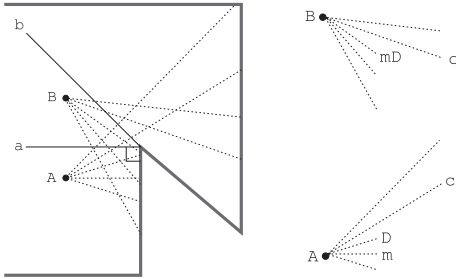


Figure 5: Ambiguous Sector with Local Minimum

**Localization Procedure.** First the robot is localized to a particular visibility sector by a cyclic string matching of the scan label computed from the sensor reading with a sector label. Next, the exact configuration of the robot in that sector is calculated by registering the scan data with known global coordinates of characteristic points in that sector. Details of this process can be found in [13].

It is possible that multiple sectors of the environment have the same label. Since our localization method cannot distinguish between such sectors, it is the global navigator’s responsibility to plan the path and select the relocalization points to that the robot is never in a situation where the current uncertainty region intersects two sectors with identical labels.

## 4 System Implementation

The mobile robot planning and navigation system described in this paper has been implemented and sup-

ports both simulated and hardware experiments. The C++ code includes modules for each of the component algorithms described in Section 3, and the navigation process described in Section 2 is implemented in the main program. The LEDA library [14] was used for low-level geometric calculations.

The input to the system is: (i) a set of one or more simple polygons, in global coordinates, representing the environment, (ii) a roadmap, and (iii) specifications of the mobile robot. In the following, we discuss some implementation details.

**Preprocessing.** During preprocessing, the visibility sectors of the environment, and their labels are computed using the following algorithm.

### CONSTRUCT SECTOR AND LABELS

1. Add the obstacle edges to the sector boundary list.
2. For each pair of the obstacle polygon vertices, identify spurious edges (defined in [10]) and add them.
3. Check intersections of the edges in the boundary list and generate sector boundary polygons.
4. Obtain center of mass for each sector.
5. Simulate range scan around the center of mass.
6. Obtain sector label using the simulated scan.
7. Compare cyclically if any two of the labels are the same. If found, store them in a same-label list.

**Component Algorithms.** Although it would be desirable to generate the uncertainty regions by compounding the ellipses based on stochastic estimation, our current implementation uses a simple min-max based estimation. This results in more conservative error regions and possibly causes us to invoke the localization process more frequently than necessary. Our current system assumes a roadmap will be provided as input, and uses straight-line local planner [2]. Once the **start** and **goal** are connected to the roadmap, the shortest path connecting them is extracted using Dijkstra’s algorithm.

The localization algorithm implementation closely follows the description in [13], subject to the modifications discussed in Section 3.3. One optimization performed in the implementation was to identify *a priori*, in pre-processing, a local maximum point M visible in each sector to be used for localization.

**High-level navigation – subgoal identification and re-planning.** After a path is extracted from the roadmap, the navigator must select a subgoal along that path as the terminus of the next trajectory command. The subgoal was selected to both avoid collision and to ensure that localization would be possible. Both these criteria were checked using uncertainty ellipses. First, the maximal prefix of the original path that is guaranteed to be collision-free is found. Next, the sectors intersecting the last uncertainty ellipse of

the pruned path are found and their labels are checked for uniqueness; if non-unique labels are found, the last path segment is removed and the process is repeated. The endpoint of the final pruned path is the subgoal.

If the first node of the path extracted from the roadmap was very close to the current start position, then we optimized the path if possible by removing that node from the planned route. This can be seen in Figure 6(b).

## 5 Experimental Results

The environment used for both the simulation and the hardware experiments is shown in Figure 2; its dimension was  $50 \times 50$  inches. The visibility sectors generated during preprocessing are shown in Figure 3, where each sector's center of mass is marked by a small  $\times$ . This environment has 71 visibility sectors, and it was found that several sectors have the same cyclic label. For instance, sector A has the same cyclic label, mMmMmMmMcDmDcmM, as sectors B, C, D and E.

**Simulation Results.** In our simulated experiments, the input range scan for the localization algorithm was generated as follows. First, a position and orientation for the mobile robot was selected at random inside the uncertainty ellipse expected to contain the robot after executing the given trajectory command, and then a simulated range scan from that point was generated.

We present two different simulated experiments. The first experiment (Figure 6) illustrates how a different route is selected when re-planning after a localization operation. The second experiment (Figure 7) is presented to enable comparison to an identical experiment presented in Section 5 conducted with our mobile robot. The first experiment uses the roadmap shown in Figure 2, while the second uses a slightly perturbed roadmap. The uncertainty parameters for the actual mobile robot were used in both cases.

In the first simulated experiment, as shown in Figure 6(a), the global planner initially selects the path `start-b-a-f-goal` by finding the shortest path in the roadmap from `b` to `f`. The node `b` is selected as the `subgoal` since the uncertainty ellipse along the segment `b-a` intersects an obstacle. After the first robot movement and subsequent localization (Figure 6(b)), the actual robot position was close enough to the node `c` so that the (re)planning by the path planner selected a different path `(new)start-c-d-f-goal`. Now, the actual `goal` is selected since the uncertainty ellipses indicate a collision-free path exists that does not encounter ambiguous sector labels. The goal is reached, within the specified tolerance, after the iteration.

Another simulation experiment is presented to enable comparison to an identical hardware experiment conducted with our mobile robot. As can be seen in

Figure 7, two iterations are expected to be sufficient to reach the goal. The first trajectory command is for the path `start-f-g`, and the second is for the path `(new)start-h-goal`.

**Hardware Experimental Results.** Our navigation system was verified using a real mobile robot in a situation identical to the second simulation experiment. The robot, the same as used in [13], has differential drive with DC gear motors and encoders (see Figure 8). Three ultrasonic range sensors are mounted on the pan/tilt head so that each sensor gives readings of  $\frac{2\pi}{3}$  radians. Only two commands — forward movement and rotate at a fixed position — were used.

The results of the experiment are shown in Figure 9. In order to get sufficiently close to the goal, three movement and localization operations were required. After the second localization, a trajectory connecting the current position to the goal is generated. The third localization confirms the arrival at the goal.

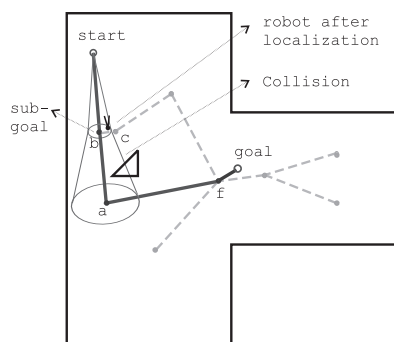
## 6 Conclusion

In this paper, we describe the design and implementation of a method for mobile robot navigation in a known indoor environment, such as a home or office, that requires only inexpensive range sensors. Our framework includes a high-level planner which integrates and coordinates path planning and localization modules with the aid of a module for computing regions which are expected, with high probability, to contain the robot at any given time. Our simulation and hardware experiments show the practicality and potential of our approach.

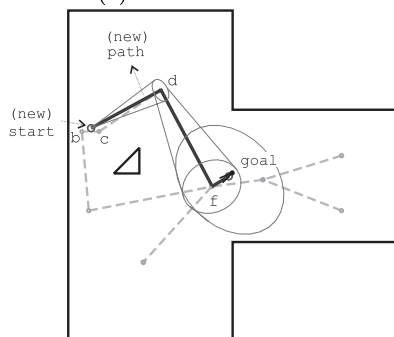
In current research, we are working to adapt the method to handle partially known environments. For instance, while the floorplans of most home or office environments may be known, there will generally be many obstacles (e.g., furniture or people) whose positions and dimensions may not be known, or might change during execution.

## References

- [1] M. D. Adams. Control and localization of a post distributing mobile robot. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 150–156, 1994.
- [2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 630–637, 1998.
- [3] S. Atiya and G. Hager. Real-time vision-based robot localization. *IEEE Trans. Robot. Automat.*, 9(6):785–800, 1991.

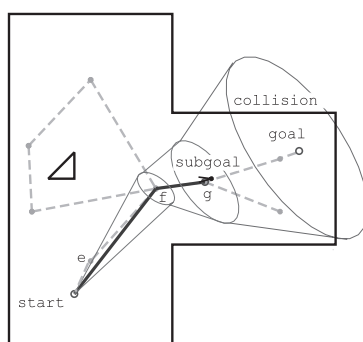


(a) First iteration.

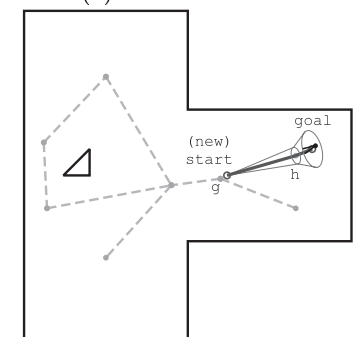


(b) Second iteration.

Figure 6: Simulation Exp #1



(a) First iteration.



(b) Second iteration.

Figure 7: Simulation Exp #2



Figure 8: Mobile robot

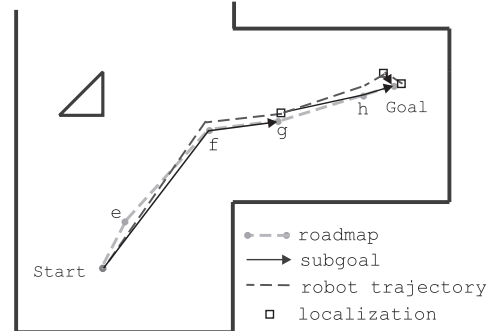


Figure 9: Hardware Experiment

- [4] N. Ayache and O. D. Faugera. Building a consistent 3-d representation of a mobile robot environment by combining multiple stereo view. In *Proc. of Int. Joint Conf. on Artificial Intelligence*, pages 808–810, 1987.
- [5] F. Chenavier and J. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2588–2593, 1992.
- [6] J. L. Crowley, F. Wallner, and B. Schiele. Position estimation using principal components of range data. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3121–3128, 1998.
- [7] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.
- [8] Gregory Dudek. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.
- [9] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robot. and Automat.*, RA-3(3):249–265, 1987.
- [10] Leonidas J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. *SIAM J. Comput.*, 26(4):1120–1138, August 1997.
- [11] K. Komoriya and E. Oyama. Position estimation of a mobile robot using optical fiber gyroscope. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 143–149, 1994.
- [12] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
- [13] S. Lee, N. M. Amato, and J. P. Fellers. Localization based on visibility sectors using range sensors. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3505–3511, 2000.
- [14] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, New York, 1998.
- [15] K. Nagatani, H. Choset, and S. Thrun. Towards exact localization without explicit localization with the generalized voronoi graph. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 342–348, 1998.
- [16] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, 1987.
- [17] A. C. Schultz and W. Adams. Continuous localization using evidence grids. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2833–2839, 1998.
- [18] C. Urdiales, A. Bandera, R. Ron, and F. Sandoval. Real time position estimation for mobile robots by means of sonar sensor. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1650–1655, 1999.
- [19] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1024–1031, 1999.