LOGnets: A Hybrid Graph Spatial Representation for Robot Navigation

Peter K. Malkin and **Sanjaya Addanki**

IBM T.J.Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Abstract

In this article we present a novel, hybrid graph spatial representation for robot navigation. This representation enables our mobile robot to build a model of its surroundings which it can then use for navigation. The models or maps that use this representation are hybrid graphs, the nodes being analogical local maps of landmark locations in the robot's environment, the arcs being the actions the robot executes to travel between the locations. This representation yields a reliable navigation tool, one which ensures that the robot can re-orient itself to recover from errors in path execution and encounters with unexpected obstacles. The LOGnet approach also meshes with human's natural approach of mapping with landmarks, instead of using angular and translational data.

1 Introduction

Real-time mapping and navigation in unknown environments are fundamental skills for an autonomous robot, skills for which many different approaches have been investigated. Given the lab shown in Figure 6, consider how a robot could map this area.

With Kuipers's Qualitative Topological Mapping scheme [4] the robot builds a topological map or graph. The nodes of this graph are "distinctive locations" where sensor values are maximized, and the arcs are the movement strategies used to travel between the locations. Although Kuipers's work was similar to our's, his simulations are simple, lacking error analysis and sensor-error modeling. Real-world performance of both mapping and navigation with this approach is unclear.

With a standard Cartesian grid-based navigation approach (*e.g.*, [1]) the robot records all the angular and distance information it measures into a grid. There are two drawbacks to this approach. One is that the robot maps everywhere its sensors can see, forcing the robot to manage data it will never actually use. Further, such grid-maps include the cumulative errors of the robot's sensors and movement mechanisms. Navigation with standard grid-maps also suffers from complex, error-prone path planning and poor failure-recovery.

The Occupancy Grid approach [5, 6] overcomes the standard grid-map's cumulative error drawback, since Occupancy Grid cell values specify the probability that the corresponding real-world locations are occupied - sensor and locomotion errors included. Occupancy Grids still contain more data than is necessary for navigation.

Davis's [3] Relational Topological Representations reduce the amount of unnecessary data retained by mapping the lab in terms of the angular and translational relationship between distinctive topological features; error bounds are included in these relations. Here, although the robot does not map everything it can see, it still uses error-prone distance and dead-reckoning measurements to define the spatial relationships between objects. Path planning and re-orientation remain complicated with this mapping scheme.

Besides these purely practical problems, metric-world spatial representations go against our natural intuition. To map a room, such as that shown, one does not record the size, shape and relative position of everything one can see; the map we make is much simpler. We would remember the landmarks of the room, and what we did to travel between these locations. Complete information about all of the intermediate areas is unnecessary until exceptions arise.

2 A Hybrid Graph Approach

We propose that the robot map its surroundings using snapshots of landmark locations and navigate using **navigation scripts**: sequences of actions that carry the robot from one landmark location to another.

The framework we have developed is a hybrid graph:

- The **nodes** of the graph being analogical local maps of distinctive locations - voting scheme versions of Occupancy Grids [5];
- The **arcs** being the sequences of actions the robot performs to travel between the locations.

We call these graphs **LOGnets.**

Thus, our mapping scheme uses no absolute distances and no dead reckoning. Further, this approach meshes with our

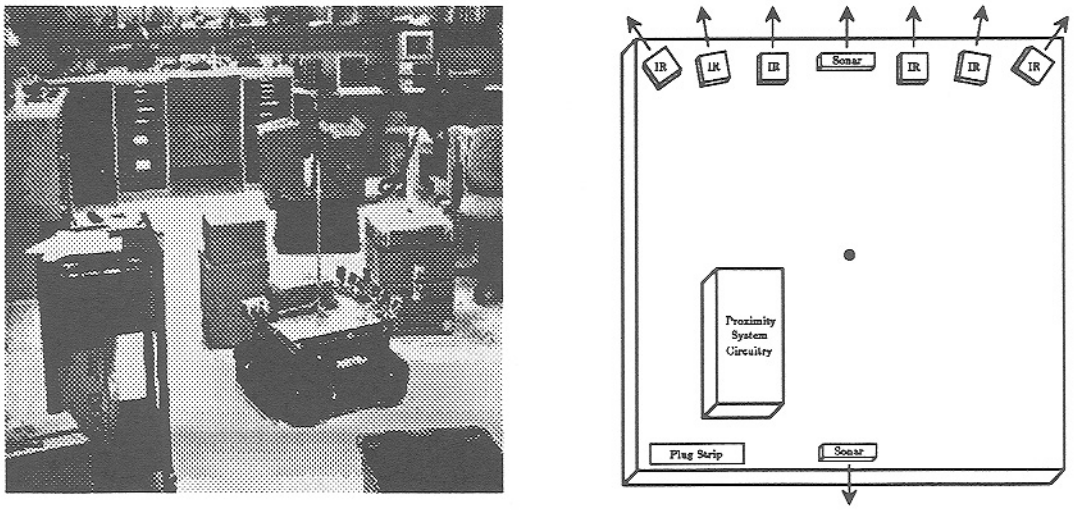

Figure 1: The LNNS System Robot

natural approach to navigation. For example, we don't use a detailed map to plan how to get to our office from the front door each morning; we just execute the set of actions that we've learned gets us to our office (*e.g.*, "Go through the door, follow the hallway to the end and then make a right into the office"). Similarly, when we give directions, we just give a series of landmarks and the actions required to get between the landmarks.

3 Our Implementation of LOGnets

The LOGnet hybrid graph representation described above is implemented in our LOGnets Navigation System, LNNS. The system's robot is a TRC Labmate Mobile Base carrying a TRC Proximity System with six infrared sensors for obstacle detection and two sonar depth sensors (Fig. 1). The robot is controlled autonomously by a Symbolics 3650 computer connected to the robot by an umbilical cord consisting of two RS-232 interface lines: one to communicate with the robot base, and one to communicate with the proximity system. The base accepts commands which set its forward and turning speeds (the forward speed for our testing = 20 cm/sec), and will return measures of how far it has turned. The proximity system accepts commands to make and return measurements, both sonar and IR.

3.1 LOG Maps

The area around the robot is represented by 36, 10 degree sectors, each sector broken into 6 depth divisions corresponding to 0.75, 1.0, 1.5, 2.0, 3.0, and 4.0 meters respectively (Fig. 2).

To make a LOG, the robot takes 4 readings with the forward and rear sonar sensors for each 10 degree sector. We do not account for sensor error, unlike [6] and [3]. A given cell value (a depth division of a sector) is incremented if the returned sonar reading is within the cell's depth limit. Hence, a cell's value indicates the confidence that the corresponding depth is accessible. Figure 3 shows an example from our experiments (the square in LOG's center indicating the area covered by the TRC robot base).

Note that a LOG is simply a 36 by 6 integer array, no actual measurement date is retained. We experimented with LOGs of fewer cells and fewer sonar readings, but found these maps were insufficiently reliable for navigation.

3.2 LOG Map Comparison

To determine if it is in a previously visited location, the robot makes a LOG of its surroundings and compares the LOG to all other LOGs in its current graph. LOG comparisons consist of three calculations, each examining a distinct type of LOG map difference. The three types of differences calculated are:

1. The total binary difference, *i.e.*, the total number of cells which disagree as to designation: unoccupied (cell value = 0), or occupied (otherwise);

2. The average root mean squared difference between the values of the LOGs' corresponding cells;

3. The average absolute difference between values of the LOGs' corresponding cells.

Two LOGs match if all three of these comparisons fall within empirically determined thresholds. Since the robot may be in the same location, but oriented differently than it was when it took the original LOG, all possible orientations of the new LOG are checked (*i.e..*, all 36 possible 10 degree offsets). Thus, if a LOG fails to match any previous LOG, the LOG does not match any previous LOG at any orientation.

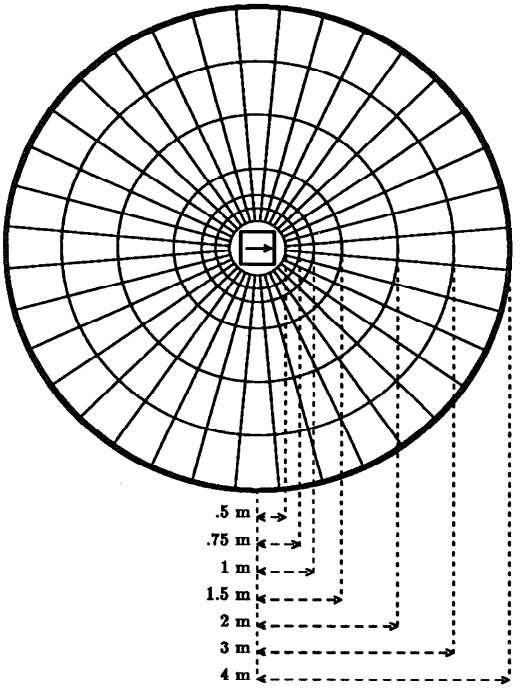

Figure 2: Local Occupancy Grid, LOG

3.3 LOGnets

The robot constructs a LOGnet by exploring its environment and measuring a LOG whenever it encounters an obstacle, an obstacle makes a location "distinctive." This may be thought of as a simplified version of Kuipers's "distinctive place," [4].

3.3.1 Subsumption Architecture-Based Rule System

The exploration the robot performs is accomplished with a subsumption architecture-based, rule system, **SABRS** (*ref.* [2]). This system consists of set a of implication rules whose antecedent conditions are triggered by the robot's sensors, and whose consequents are **action symbols** corresponding to the possible actions the robot's mechanisms can perform. This wandering system operates by continually executing the following sequence of actions henceforth referred to as a *step*:

1. The robot's sensors are queried and the data from them is passed to the production rules.
2. Every rule whose antecedents are satisfied asserts its consequents.
3. All asserted consequents are sorted by the subsumption architecture which determines which of the asserted robot actions to actually implement (Fig. 4).

The robot's wandering is just the repeated execution of this rule-firing sequence. The *cost* of an arc is simply the number of steps in it.

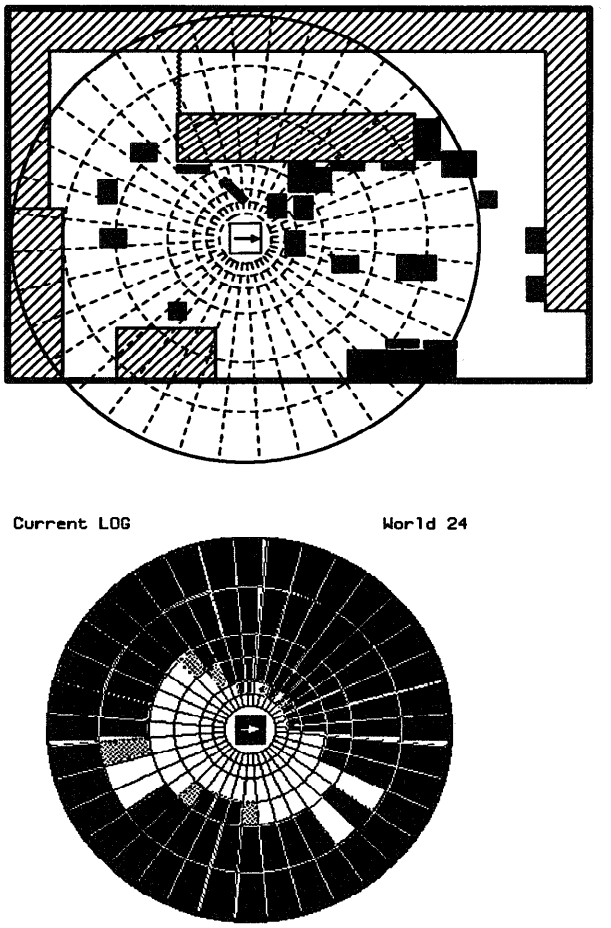

Figure 3: Layout 8 LOG 24

Since these sensor readings are used without additional calculation and since the actions the robots performs are simple, these steps are performed very quickly (less than 1 Hz, .5 Hz. of which is spent waiting for the proximity system to respond). Thus, the robot is able to react to its environment in real time. Further, since the SABRS assertions are made in form of primitive symbols (*e.g.* `stop` and `turn-left`) the SABR system acts as a real-time signal/symbol bridge. Lastly, new SABRS rules can easily be added to get the robot to perform more complex behaviors.

3.3.2 LOGnet Construction

When an obstacle is detected, the robot measures a LOG. If this is the first location visited, the new LOG is simply added to the LOGnet. Otherwise, the robot compares the new LOG with all other LOGs in its LOGnet. If the new LOG does not match any other LOG, a new node is added to the LOGnet with an arc connecting the LOG representing the location the robot last visited to the new LOG, the new arc containing

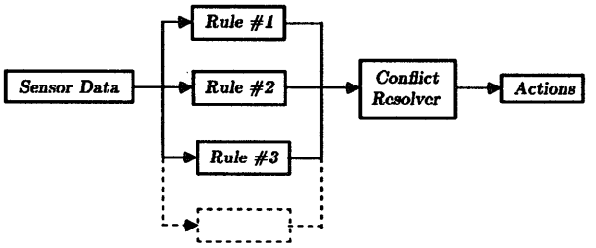

Figure 4: Subsumption Architecture-Based Rule System

the actions (represented by action symbols) that the robot executed in traveling between the two locations. One such arc might be:

```
stop, turn-left,
stop, turn-left,
go-forward, no-turn,
go-forward, no-turn,
go-forward, no-turn
```

Suppose, the robot travels from the location represented by LOG 1 to a new location where it measures LOG 5, and that the sequence of actions the robot executes in getting between the two locations is called Arc A (Fig. 5). If LOG 5 does not match any other LOGs in the current LOGnet, LOG 5 is connected into the LOGnet with Arc A from LOG 1.

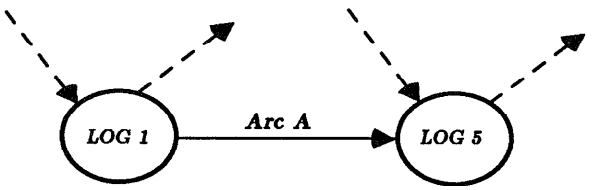

Figure 5: Linked LOGs

If the new LOG does match a LOG in the robot's LOGnet, it means this is a location visited previously. If the current LOGnet lacks an arc connecting the current LOG to the LOG of the previous location, the new arc is added to the LOGnet to indicate that this path exists. If an arc already exists, then the costs of the new arc and the existing arc are compared, the cheaper arc stored in the LOGnet.

Note that the LOGnets the robot produces are not necessarily complete mappings of its surroundings - there may be places the robot does not map simply because its rule-based wandering never gets it there. The topic of complete mapping is beyond the scope of this paper.

3.4 LOGnet Navigation

To navigate, the robot first orients itself by wandering until it finds a location it recognizes (through LOG comparisons). The robot then asks the outside observer for a goal location.

Given this goal location, the robot calculates the lowest cost path to the requested location by referring to its LOGnet, this path calculation requiring only simple graph tracing. To execute the calculated path, the robot carries out exactly the actions the path's arcs dictate. Note that the subsumption architecture wandering system is still in effect to the extent that the robot will avoid any unanticipated obstacles.

The robot checks it is on course by verifying that it reaches the locations indicated by its path. If the LOGs measured match the path's LOGs, the robot continues. If a LOG fails to match the path's LOG, matching another LOG instead, the robot recovers by planning the least expensive path to the goal from its current location. If the current LOG fails to match any LOG in the LOGnet, the robot wanders until it finds a LOG it recognizes, calculating a least-cost path to the goal-location when it does.

Thus, using LOGnets, the robot can recover from unexpected obstacles, the subsumption rules ensuring it avoids collisions. The robot can also recover from path execution errors by determining paths to its goal from wherever it finds itself.

4 Experimentation

In one of our tests of the LNNS LOGnet Navigation system, we constructed eight different environments in our lab, changing the room layout by moving real-life obstacles (*e.g.*, chairs, trash cans and equipment carts)(Fig. 1). In each of these environments we had the robot build a LOGnet of the room, and then use this LOGnet to reach positions we specified.

4.1 LOGnet Construction

We had the robot map several of the room layouts multiple times. In each given environment the robot chose approximately the same landmark locations, demonstrating that these positions are stable. Further, since the robot was started in different locations, the position of the landmark locations is independent of where the robot begins - the position of the landmark positions dependent solely on the robot's wandering system and the sensitivity of its sensors.

Figure 6 shows one of our test environments with the path the robot followed in constructing its LOGnet. It took the robot approximately 2 hours to perform this exploration.

4.2 LOGnet Utilization

Figure 7 shows the path the robot took to get from location#26 to #28 using the LOGnet described above. The robot first wandered until it encountered an obstacle. It measured a LOG and correctly determined its location to be location #26. It then asked where it should go and we specified location #28.

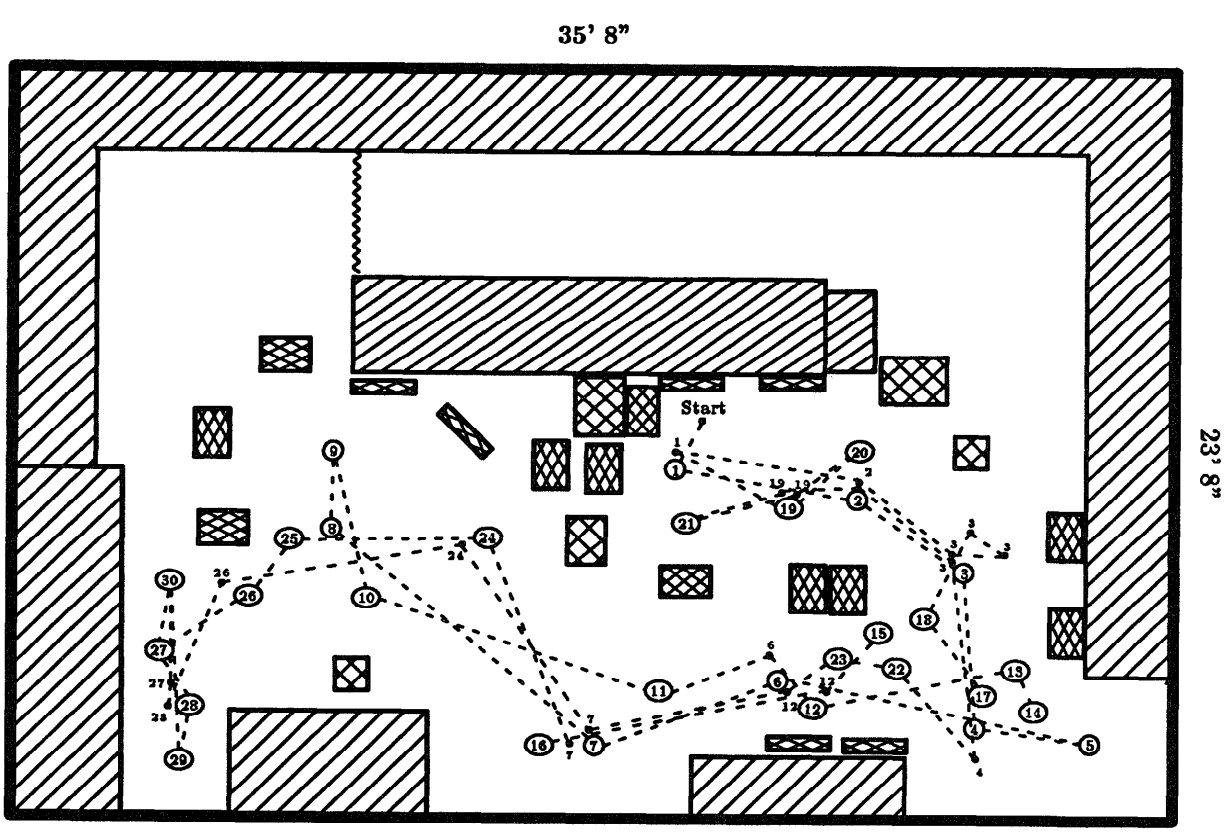

Figure 6: Layout 8, Mapping 2

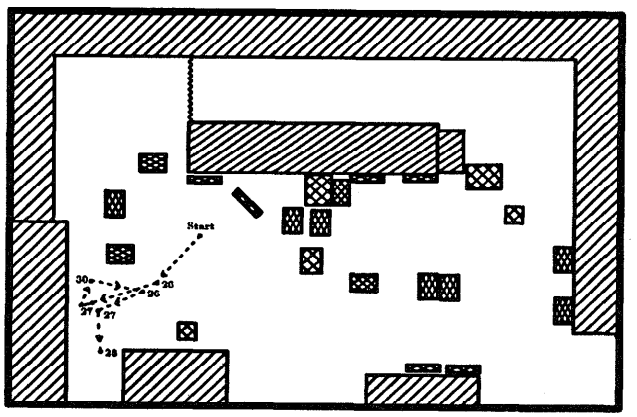

Figure 7: Test #2, Layout 8,Mapping 2

Given its starting location, LNNS calculated that to get to location #28 from #26, the robot should go from location #26 to #27 to #28. Although the robot successfully navigated to location #27, it failed to get to #28, arriving instead at location #30. Recovering from this path execution error, LNNS determined that it should go from #30 to #27 to #28. The robot again failed to reach the desired goal (location #27), traveling instead to location #26. LNNS then calculated the best path to goal location #28 being from #26 to #27 to #28. This time the robot successfully traveled from #26 to #27 and then from #27 to #28.

4.3 Discussion of Results

There are several things to note about the way the robot constructed is LOGnet. The first is that the robot always recognized locations it had visited previously despite translational offsets (up to 2 feet) and rotational offsets (0 - 360 degrees), locations #4 and #3 for example.

One can also see that the robot traveled between landmark location along particular routes (*e.g.*, between locations 2 and 3). It is these paths that the robot's LOGnet arcs roughly specify.

Certain areas in the room required more LOGs than others (*e.g.*, the area around locations 13, 14 and 17 compared to that around location 3). This is due to the robot's sonar sensors. Because of their low bandwidth, if the robot is positioned near the corner of a box, the sonar may not sense the box because the sonar signal is deflected away. Hence, if the robot is near any form of corner, the LOGs are more sensitive to translational offsets.

As demonstrated in the navigation test described above,

LNNS was able to recover from errors in path execution. This recovery was quick since the graph-tracing method of determining the best solution path from wherever the robot finds itself is as simple and quick as the method for planning the initial path.

In the test above the robot correctly identified all of the locations it visited. There were occasions during our testing when the robot incorrectly identified its location, but this happened less than 5% of the time. Such errors were overcome because when the robot attempted to follow its calculated path to the goal location, it would correctly identify the next world it encountered, thereby re-orienting itself.

Through experimentation we learned that a LOG sector angle of 10 degrees yielded reliable landmark location identification and path tracing. When we tried using greater LOG sector angles (*e.g*, 20 degrees) we discovered that although the robot was able to correctly identify its location, it was not able to orient itself accurately enough to trace its paths.

In cases where the robot encountered unexpected obstacle (*e.g.*, human co-workers), the robot easily recovered by stopping and measuring a LOG. This LOG would not match any in the robot's LOGnet (since the robot was never forced to stop in this position before), and so the robot would wander until it encountered a location whose LOG it did recognize.

5 Future Work

Higher level actions would allow the robot to track its surroundings while executing a path. Hallway following, for example, could be accomplished simply by steering the robot so as maintain the same distance measurements from side-facing sonars. Thus, the robot could adjust its path mid-course to coincide more closely with the path being followed, greatly increasing the reliability of the robot's path following.

Higher level actions would also compress path specifications. With the concept of hallway-following, a path could specify that the robot go down a hallway until the first intersection, the robot executing this path by using its hallway following rules until the hallway ends (*i.e.*, at the first intersection.

The LOGnet itself could be used to disambiguate two matching LOGs. To verify that it is really in the corresponding previous location, the robot could perform the actions of an arc from the supposed current location and verify that it reaches the location represented by the connected LOG. A similar technique is proposed in [4].

Our implementation's reliability could also be increased if LOGs contained information about the identity of the obstacles surrounding the robot (*e.g.*, the robot is facing an elevator). By checking for the existence of particular objects, similar locations could be distinguished from others that look identical as seen by the sonars.

6 Conclusion

LNNS, the LOGnet Navigation System demonstrates that the hybrid-graph spatial representation described above provides a compact and robust spatial representation for navigation. LOGs, analogical, local depth maps, allow the robot to reliably identify its location, and the SABRS Subsumption Architecture-Based Rule System provides the real-time, signal-symbol interface that links the landmark locations together procedurally.

The LOGnet representation avoids the cumulative errors associated with non-incremental metric spatial representations and enables the robot to recover from path execution errors quickly and effectively. The hybrid-graph spatial representation also fits human's natural approaches to navigation, relying on landmarks and procedural information rather than on absolute distances and dead-reckoning.

Acknowledgments

We would like to thank Norman Haas, Alberto Elfes and Paul Viola for their contributions to this work.

References

[1] Nilsson, N. A Mobile Automaton: An Application of Artificial Intelligence Techniques. In Proceedings of the International Joint Conference of Artificial Intelligence, 1969, 509-520.

[2] Brooks, R. A. A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control. In Proceedings of the IEEE International Conference on Robotics and Automation, 1987, 106-110.

[3] Davis, E. *Representing and Acquiring Geographic Knowledge*. Los Altos, California: Morgan Kauffman Publishers, Inc., 1986.

[4] Kuipers, B. J. and Y. Byun. A Robust Qualitative Method for Spatial Learning. In Proceedings of the AAAI-88 Seventh National Conference on Artificial Intelligence, August, 1988, pp. 774-779.

[5] A. Elfes. Sonar-Based Real-World Mapping and Navigation. In IEEE Journal of Robotics and Automation, v. RA-3, n. 3, June 1987.

[6] A. Elfes. A Tesselated Probabilistic Representation for Spatial Robot Perception and Navigation. In Proceedings of the 1989 JPL/NASA Conference on Space Telerobotics, January 31 - February 2, 1989, JPL, Pasadena, CA.