Global Symbolic Maps From Local Navigation

David P. Miller
Jet Propulsion Laboratory
MS 301-440, 4800 Oak Grove Drive
Pasadena, CA 91109
(818) 354-9390, dmiller@ai.jpl.nasa.gov

Marc G. Slack
MITRE Corporation
7537 Colshire
McLean, VA 22102
(703) 883-5518

Abstract

In order to navigate autonomously, most robot systems are provided with some sort of global terrain map. To make storage practical, these maps usually have a high-level symbolic representation of the terrain. The robot's symbolic map is then used to plan a local path. This paper describes a system which uses the reverse (and perhaps more natural) process. This system processes local sensor data in such a way as to allow efficient, reactive local navigation. A byproduct of this navigation process is an abstraction of the terrain information which forms a global symbolic terrain map of the terrain through which the robot has passed. Since this map is in the same format as that used by the local navigation system, the map is easy for the system to use, augment, or correct. Compared with the data from which the maps are created, the maps are very space efficient, and can be modified, or used for navigation in real-time. Experiments with this system, both in simulation, and with a real robot operating in natural terrain, are described.

Introduction

Traditional robot systems often follow the architecture shown in Figure 1 when performing navigation. The data from the world model flows strictly outward. The entire sense-plan-act cycle is often only performed once, and an entire path is planned for the robot to achieve its next goal. This architecture makes several assumptions which we have found during experimentation, are not often valid [Miller89, Gat90, Miller90b]. In particular, such systems assume that the global map is correct and inviolate, and that the robot has sufficient knowledge and accuracy, and the world is sufficiently predictable, that the robot can execute a complicated path open loop. Such assumptions are seldom justified.

Most work to correct these assumptions has concentrated either on the global map making, or on the robot's path execution. The reactive approach to mobile robots [Brooks86, Connell87, Arkin89, Miller90a, Payton88] are examples of the latter. Examples of the former include [Davis86, Elfes87]. Both of these sys-
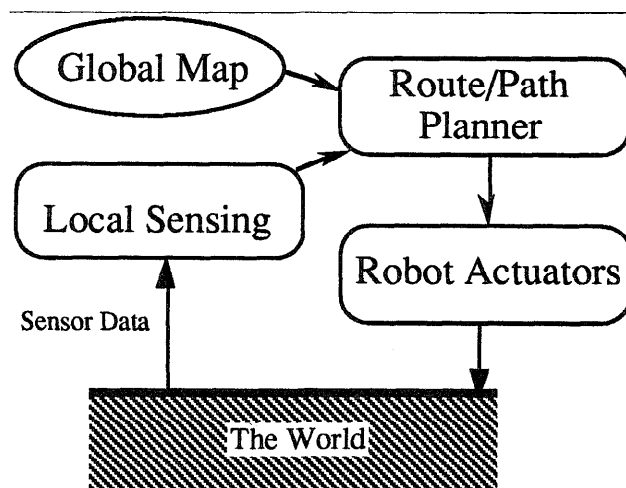

Figure 1: Traditional Robot Planning Architecture

tems describe the process of global map formation, but neither address how their maps can be used for navigation; Davis assumes an idealized human vision sensor, while Elfes' system requires very large amounts of storage.

Proponents of the reactive approach state that no global map is necessary, that in fact it is detrimental to successful navigation in many cases. Their argument generally follows the lines that such a map is never completely accurate, and therefore simply propagates incorrect information forever. Their work has empirically demonstrated that no global map is necessary to safely move about locally in the world [Brooks88, Miller90a].

However, most actual applications of mobile robots demand that the robot do more than move about. Delivery, inspection, and even exploration tasks usually require that the robot be able to go to a specific place in the world, or at least move in a specific direction. To do these efficiently, some sort of map is very helpful.

While we believe that the reactive approach is both necessary and sufficient to handle immediate local nav-

igation, a global map is necessary to allow a robot to move efficiently when its goal lies beyond its sensory horizon [Miller90b, Andress88].

However, it is necessary to have the right global map. The map must be in a form that is useful for local navigation. The robot's sensors are usually adequate to keep it from bumping into an obstacle, but are inadequate for deciding which way the robot should go around the obstacle. One direction may lead to a dead-end, requiring significant backtracking. This is the information that the robot must be able to quickly extract from the global map. For these reasons the map must be space efficient, easy to retrieve, must be easily updated (so that it remains useful), and should be interpretable by other systems, including humans (while this is not strictly necessary, it makes experimentation easier and the results more meaningful).

This paper presents a local navigation system based on navigation templates or NaTs. These templates are used to abstractly represent the goals, obstacles, waypoints etc. in the robots world. Each template has a gradient field. These fields can be quickly combined to calculate the robots correct trajectory for its current position. One of the unique features of the NaT navigation system, is that once the NaTs are created from the sensor data, they make a compact and easy to use representation for building global maps. Since the NaTs are used for calculating the local path of the robot, they are inherently useful for longer range path planning as well. The remainder of the paper describes NaTs, experiments we have performed with this system, and the value of the global map that we get virtually for free while doing local navigation.

Navigation with NaTs

The general problem of navigation involves reasoning about both global and local issues. Global navigation deals with planning a route that avoids major obstacles, and efficiently leads to the goal. The direction to travel around a major obstacle is a global issue. At the local level a navigation system must be intimately connected to the physical world and reason about the robots physical relationship to the objects in the world as those objects relate to the current navigation task.

Navigation Templates are primitive building blocks for constructing highly flexible navigation plans which capture the essence of the navigation situation (i.e., the task and relevant environmental constraints). There are two types of Navigation Templates: those which are used to characterize the basic local navigation task being pursued, and those used to model known environmental constraints and characterize the relationship of the constraints to the navigation task. Once a navigation plan has been built from a set of Navigation Templates, a powerful heuristic is employed to isolate the currently critical aspects of the plan and quickly (a few times a second) generate guidance for the robots low level control system. As time passes and/or the robot moves through the world, causing changes in the robots perception of the world, the navigation plan must be incrementally updated in order to remain useful. The fact that Navigation Templates do not depend upon one another allows them to be quickly translated, rotated, scaled, inserted and/or deleted from the navigation plan without affecting the other templates. Thus, a Navigation-Template-based navigation plan can be quickly modified in order to remain synchronized with the robots perception of the world.

Substrate NaTs

Substrate Navigation Templates (or s-NaTs) describe a particular navigation task or goal. Each s-NaT defines a gradient field indicating at every position in space the direction of travel that best serves the navigation task being represented. For example, if the current navigation task is to move up a hill, then the gradient would, at every position, be directed to the top of the hill. It is important to note that the substrate is defined independently of the environmental constraints which limit the robot's ability to follow the substrate's gradient field. As a result, the gradient field of an s-NaT is just as likely to direct the robot through a wall as it is to direct the robot to avoid walls. While it is possible to create an s-NaT with almost any gradient field, this work has identified three types of s-NaTs which can be used to characterize a large class of navigation tasks: direction s-NaTs, position s-NaTs, and trajectory s-NaTs.
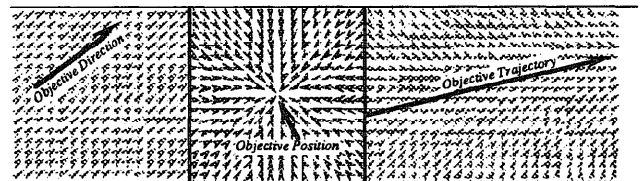

Figure 2: Substrate NaTs

A direction s-NaT is used to describe navigation tasks such as "head west", "move across the room", or "walk through the field", in which lateral deviations with respect to the direction are permitted. The gradient field for a direction s-NaT is trivially defined at every location to point in the objective direction (see Figure 2). To describe tasks such as "go over to the car" or "move to the rock", a position s-NaT is used. The gradient field for a position s-NaT is defined at every position to point toward the objective position (see Figure 2). The final type of s-NaT is a trajectory s-NaT, where the gradient converges to an objective trajectory through space. This s-NaT is useful for characterizing navigation tasks such as: "proceed up this hall", "follow this path", or "drive in this lane " (see Figure 2).

Modifier NaTs

S-NaTs provide a basic scheme for accomplishing a navigation task in the form of a gradient field. Because the world is full of obstacles that place restrictions on the robot's ability to move through the world, the s-NaT's basic scheme for accomplishing a particular navigation task will need revision in order to accommodate environmental constraints. Modifier navigation templates (or m-NaTs) are used to model environmental constraints as they relate to the current navigation plan. An m-NaT can be created for any convex geometric object; concave objects are represented using multiple m-NaTs. For example, a trajectory s-NaT can be used to provide a basic scheme for moving across a field, while m-NaTs would be used to model the relationship between relevant environmental constraints, such boulders and trees, and the navigation task. Together the s-NaT and the m-NaTs are used to construct a navigation plan.

Central to building a navigation plan is the notion that there are two ways to move around an obstacle: clockwise and counter-clockwise. Thus, each of the m-NaTs has an associated spin, indicating in which direction around the obstacle the navigation plan dictates the robot proceed. Determining the spin of an m-NaT is typically accomplished through a simple analysis of the way that the obstacle relates to the robot and the current s-NaT.

Trajectory Calculations

S-NaTs are used to characterize a navigation task and to provide a basic scheme for accomplishing that task (i.e., its gradient field). M-NaTs are used to model environmental constraints as well as their relationship to the navigation task (i.e., their spin). Together an s-NaT and a number of m-NaTs are used to construct a rough navigation plan for accomplishing a given navigation task. The resulting gradient is intended to serve as run-time guidance for the robot's low- level control loop. To transform the qualitative NaT-based plan for accomplishing a task into quantitative guidance for the robot's actuators a transformation function is provided. The transformation function calculates the preferred direction of travel from a position which satisfies the constraints imposed by the NaTs. Details of the trajectory calculation are given in [Slack90]. Figure 3 shows a simulated run of a robot following the result of the trajectory calculation as applied to the shown NaT-based plan. Note that the trajectory computation was only performed at the robot's location, and that the additional vectors are included only for illustrative purposes.

Extracting Terrain Features

To form the basis for local navigation the robot must transform the vast amount of incoming sensor data into a symbolic representation of its local surroundings.
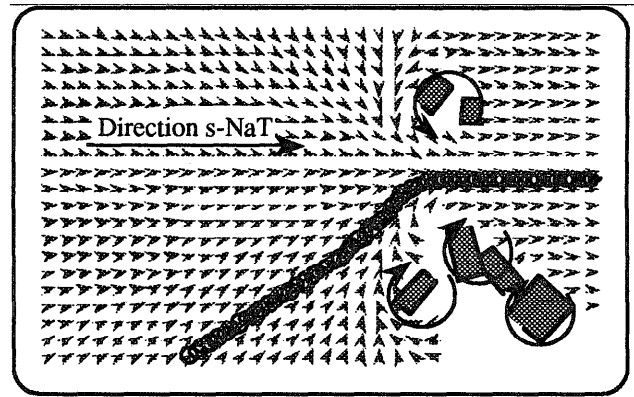

Figure 3: Simulated Run Through NaT Field

The robot is provided with a sparsely filled and dynamically changing height map of its local surroundings. From this data relevant local features are extracted. Because the local features are linked to a global position in the world, as the robot moves, and more features come into view, the robot's global perception of the world is incrementally increased.

Although some information is lost, the resulting global map is a highly compact representation of the robot's sensor readings and serves as the robot's memory of the terrain through which the robot has passed. Compaction of the robot's sensor input allows the robot to remember vast portions of the space through which it has traversed (thousands of kilometers), as opposed to the relatively small amount which could be remembered if such compaction were not performed (tens to hundreds of meters). In addition to these benefits, the global map supports the navigation system's need for a symbolic understanding of the terrain. Global navigation is supported by the global map, in the event that the robot must return to a portion of the terrain through which it has already passed.

Local navigation is supported by the global map, in the event that local features which become occluded from the robot's sensors can be accessed and paths which may lead to the robot into a dead end can be avoided. The robot's sensor data is generated by transforming the range data acquired by the stereo vision system into unevenly spaced position and elevation data. This information is then stuffed into a terrain grid where the cells of the grid contain quantities, such as the elevation, slope, and roughness [Wilcox87].

Terrain features are extracted from the terrain grid by running a filtering function over selected portions of the terrain grid (see Figure 4). The result of the filtering function is stored in the observation slot of the feature so that the value can be accessed by the navigation system at a later time. Because of the way filters work, there will usually be several terrain features associated with each feature in the real world.
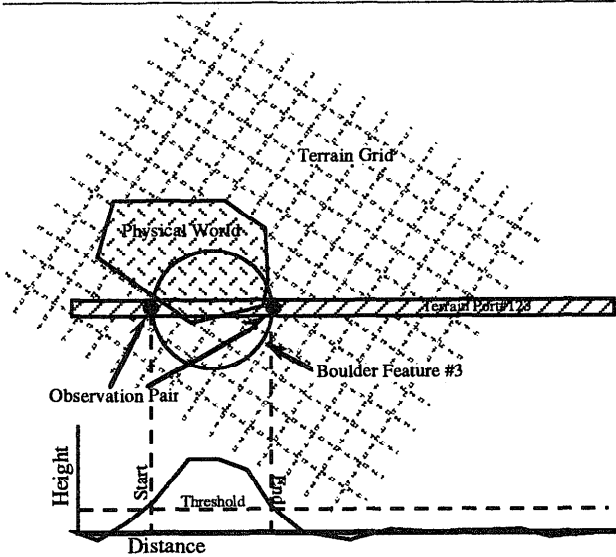
Figure 4: Extracting Terrain Features

These features need to be merged. There are several reasons for merging overlapping terrain features. Some are related computational or space efficiency requirements. Another is that these features are eventually used to form m-NaTs. Overlapping m-NaTs all have to have the same spin, or else the robot will not be able to derive a coherent trajectory. Events are simplified by merging features into a single feature (and a single resultant m-NaT) when the features overlap heavily (see Figure 5).
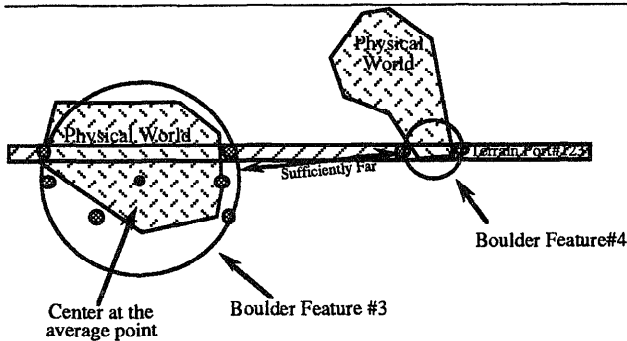

Figure 5: Combining Terrain Features

Experimental System

The system described above has been extensively tested in simulation, and several runs have been done using the JPL Planetary Rover Navigation Testbed. The testbed is a six-wheeled, three-bodied high-mobility vehicle. All actuators and joints are instrumented and the vehicle has a precision gyro-compass, and a pan-tilt head with several video-cameras. All

computation and power are onboard. The vehicle masses a little more than a ton. All experiments were run in the Arroyo outside of JPL. The terrain there consists of rough sand and dirt with ridges, rock outcroppings, bushes, and scattered boulders.

During the experiments, the robot was given a goal to get to that was some distance away (typically one-hundred meters). The goal was specified by absolute coordinates. The rovers position and orientation were also provided to the robot in the same coordinate system. The robot would then operate in a completely autonomous mode. There was a single datalink back to the support vehicle so that data could be logged.

The robot started out by taking a visual stereo panorama of the terrain in front of it. The range map was converted into a height map [Matthies91]. A filter which overlayed a slightly inclined ground plane over the absolute value of the height data was used to extract significant terrain features. Overlapping features were combined and turned into m-NaTs. The goal was modeled by an objective position s-NaT. The m-NaTs were spun depending on the robot's orientation, and the NaT's position relative to a line between the robot and the goal. The immediate trajectory was calculated and the robot would move in that direction. Approximately every two meters, the robot would take and process new images, and recalculate its trajectory direction.
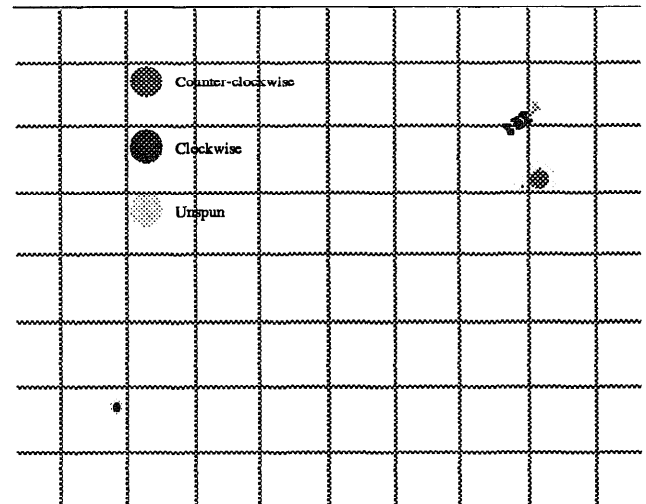

Figure 6: Start of the Arroyo Run

Figure 6 shows the robot after its first two meter move. The goal is in the lower left-hand corner. A small group of rocks was spotted by the initial panorama off to the robot's left. Figure 7 shows the robot after it has traversed approximately thirty meters. To the robot's immediate right is a NaT spun clockwise. Ahead of the robot about eight meters is a NaT spun in the opposite direction. Several terrain features that the robot has seen on its way have had

their NaTs despun. These objects are too far behind the robot to influence its trajectory calculation. By limiting the range that NaTs can influence the trajectory calculation we ensure that the calculation can be performed quickly. Should the robot's travels bring these objects back across its path, they will be respun in whatever is the appropriate direction for the robot's current task.
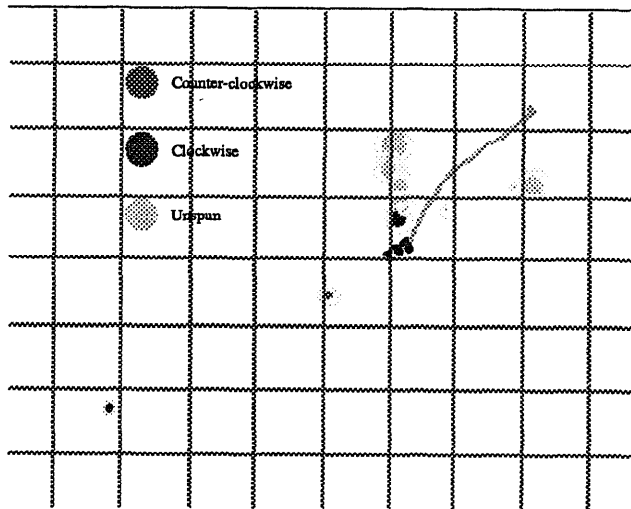

Figure 7: Thirty Meters into the Run

An object observed by the vision system has a limited lifetime. If the same area of space is viewed a second time, then the object will quickly be removed from the global map. Therefore, if a person walks across the robot's path, the robot will attempt to avoid the person, but it will not mark the person's location indelibly in its memory. If the robot views terrain features overlapping the same spot several times, then those object will be toggled to permanent status, they can then only be removed by being observed not to be there several times. In all observations (whether positive or negative) a terrain feature does not have to appear in exactly the same spot several times. It is only necessary that the feature's "center of mass" overlap the previous feature. In these ways, the system is resistant to sensor noise and sensor errors.

Using the Global Map

The system described above was created primarily to get a robot from one location to another through previously unknown terrain. Yet even in this situation, a global map has proved very helpful.

Global Maps for Traverse Efficiency and Safety

The map allows the system to remember the terrain features it has just passed. Because of the position of the camera on the robot, iy cannot see the ground

closer than two meters in front of the robot. The global map keeps the robot from turning into boulders which are along side of it, but out of sight.

We have also performed a set of simulated speed runs with multiple goals [Gat91]. In these situations, the robot takes images, extracts features and modifies its plan while on the move. As the robot goes from goal to goal, it often crosses near previously viewed terrain. If an obstacle forces the vehicle to make a sharp turn, then it must be able to get terrain data outside of its field of view. By using the data in the global map, the robot can often avoid having to image areas which might otherwise require imaging. This allows the robot to stay in constant motion, rather than stopping due to the computational expense of large amounts of image processing. The robot only has to image the area that it is actually passing through, and can rely on previous imaging to cover immediately adjacent areas (which are necessary to ensure safe navigation).

Maintaining Maps Over Long Traverses

Each of our experiments have involved on the order of a hundred meters of travel. During traverses of this length, dead-reckoning from the robot's wheel encoders has proven sufficient to keep the terrain features lined up correctly in the global map. However, a global map's greatest value occurs when traverses are much larger (and the potential penalties for making a wrong turn are also much greater). In traverses much greater than a hundred meters, dead-reckoning will be insufficient to keep reality coordinated with the global map.

The terrain features are linked into the global map by the position at which they occur (in the global terrain grid). It is assumed that an occasional terrain matching process is performed that matches the robot's position into the global map. After the initial terrain matching, other methods such as dead reckoning and feature matching can be used to maintain the robot's position in global coordinates. The original terrain matching can either be performed by the sensing system using a match on terrain grids [Wilcox87], or by matching landmark features using a method similar to that described in [Davis86]. In the latter, the global terrain grid would be preprocessed, and have terrain features extracted. The pattern of these features would be compared to the pattern of features extracted form the locally observed terrain grids. The robot's position would be determined by coordinating the two patterns of features. Because of the great reduction of points which must be matched, as compared with the raw terrain elevation maps, matches at this symbolic level require significantly less computation.

It is important to realize that the positioning of terrain features in the global map would not be exact. If terrain matching is not done often enough, or is not accurate enough, or if the robot covers a large stretch of relatively featureless terrain, it is possible for features to be misplaced. Rather than placing the features into

the map with global coordinates, they should be linked together with uncertainty bounds that can be updated with new observations [McDermott84, Davis86]. However, to do so would lose much of the computational advantage that is gained by using the symbolic information (in featureless landscapes this problem arises with most any other matching scheme short of a global positioning system). When tying feature positions to global coordinates, the system gets the computational advantage of reduced sensing, and quick insertion and access of features. However, only major features should be put into the map - to reduce the possibility of an inconsistent positioning.

A final concern about maintaining a global map is space efficiency. The maps for this system require little storage. The terrain features, once combined, are just a few data slots each. Most of these slots are needed only for performing terrain matching, and are not necessary for forming the NaTs. For strictly navigational purposes, those fields can be deleted. While very space efficient, these maps are not an appropriate structure for storing truly huge areas filled with features. For global maps of areas on the order of square kilometers, multiple maps organized in some hierarchical structure such as a quadtree may be more appropriate.

Conclusions

The NaT-based local navigation system has proven quick and reliable for navigation through outdoor terrain. As a side effect of this system, a global obstacle map is created which can aid in long range planning. By extracting the global map from the local navigation system, we ensure that the data in the map is timely. The robot can also be assured that critical data marked in the map is accurate; that what is marked as an open space was an open space at the time that the robot passed through it, because the robot did in fact pass through it. Global maps are especially useful in terrain with dead-ends or costly detours. The map that is produced is exactly the kind of map that can be efficiently used by the local-navigation system. The map is easy to update, and with occasional terrain matching, can be kept quite accurate.

References

[Andress88] Andress, K., M., Kak, A., C., Evidence Accumulation & Flow of Control in a Hierarchical Spatial Reasoning System, in the AI Magazine, Volume 9, Issue 2, Summer, 1988, or Technical Report 88-9, School of Electrical Engineering, Purdue University, 1988.

[Arkin89] Arkin, R., C., Three-Dimensional Motor Schema Based Navigation, in the Proceedings of the NASA Conference on Space Telerobotics, JPL Publication 87-7, Pasadena, California, Jan 31 - Feb 2, 1989.

[Brooks86a] Brooks, R., A., A Robust Layered Control System for a Mobile Robot, in the IEEE Journal of Robotics and Automation, Volume RA-2, Number 1, pp. 14-23, March 1986.

[Brooks88] Brooks, R., A., A Robot that Walks: Emergent Behaviors from a Carefully Evolved Network, MIT Technical Report, September, 1988.

[Connell87] Connell, J., H., Creature Design with the Subsumption Architecture, in the Proceedings of the International Joint Conference on Artificial Intelligence, Milano, Italy, 1987.

[Davis86] Davis, E., Representing and Acquiring Geographic Knowledge, Morgan Kaufmann, 1986.

[Elfes87] Elfes, A., Sonar-based Real-World Mapping and Navigation, in the IEEE Journal of Robotics and Automation, Volume RA-3, Number 3, pp. 249-265, June 1987.

[Gat90] Gat, E., Slack, M., G., Firby, R., J., Miller, D., P., Path Planning and Execution Monitoring for a Planetary Rover, in the Proceedings of the 1990 IEEE Conference on Robotics and Automation, 1990.

[Gat91] Gat, E., Reliable Goal-Directed Reactive Control of an Autonomous Mobile Robot, Virginia Tech Department of Computer Science, Ph.D. May 1991.

[Matthies91] Larry Matthies, Stereo vision for planetary rovers: stochastic modeling to near real-time implementation, JPL D-8131, January 1991.

[McDermott84] McDermott, D. V., Davis, E., Planning and Execution Routes Through Uncertain Territory, Artificial intelligence, Volume 22, pp. 107-156, 1984.

[Miller89] Miller, D.P., Execution Monitoring for a Mobile Robot System, in the Proceedings of the 1989 SPIE Conference on Intelligent Control and Adaptive Systems, vol 1196, pp. 36-43, Philadelphia, PA, November 1989.

[Miller90a] Miller, D.P., Rover Navigation Through Behavior Modification, in The Proceedings of the Space Operations Automation and Robotics Workshop, NASA, Albuquerque, NM, June 1990.

[Miller90b] Miller, D.P. & Gat, E., Exploiting Known Topologies to Navigate with Low-Computation Sensing, in The Proceedings of Sensor Fusion III, SPIE 1990 Cambridge Symposium, Cambridge, MA, November 1990.

[Payton88] Payton, D., W., Internalized Plans: A Representation for Action Resources, in the Proceedings of the Workshop on Representation and Learning in an Autonomous Agent, Lagos, Portugal, November, 1988.

[Slack90] Slack, M.G., Situationally Driven Local Navigation for Mobile Robots, Virginia Tech Department of Computer Science, Ph.D. May 1990.

[Wilcox87] Wilcox, W., H., Gennery, D., B., Mishkin, A., H., Cooper, B., C., Lawton, T., B., Lay, N., K., Katzmann, S., P., A Vision System for Mars Rover, in the Proceedings of SPIE's 7th Conference on Mobile Robots II, Volume 852, 1987.