PERCEPTION FOR MOBILE ROBOT NAVIGATION: A SURVEY OF THE STATE OF THE ART

David Kortenkamp
The MITRE Corporation
1120 NASA Road 1
Houston, TX 77058
korten@aio.jsc.nasa.gov

ABSTRACT

In order for mobile robots to navigate safely in unmapped and dynamic environments they must perceive their environment and decide on actions based on those perceptions. There are many different sensing modalities that can be used for mobile robot perception; the two most popular are ultrasonic sonar sensors and vision sensors. This paper examines the state-of-the-art in sensory-based mobile robot navigation. The first issue in mobile robot navigation is safety. This paper summarizes several competing sonar-based obstacle avoidance techniques and compares them. Another issue in mobile robot navigation is determining the robot's position and orientation (sometimes called the robot's *pose*) in the environment. This paper examines several different classes of vision-based approaches to pose determination. One class of approaches uses detailed, *a priori* models of the robot's environment. Another class of approaches triangulates using fixed, artificial landmarks. A third class of approaches builds maps using natural landmarks. Example implementations from each of these three classes are described and compared. Finally, the paper presents a completely implemented mobile robot system that integrates sonar-based obstacle avoidance with vision-based pose determination to perform a simple task.

1 INTRODUCTION

In order for mobile robots to navigate safely in unmapped and dynamic environments they must perceive their environment and decide on actions based on those perceptions. Although there are many different ways to sense the world, most robots use ultrasonic sonar sensors and visions sensors. Typically, mobile robots use sonar sensors to avoid collisions with objects in their environment and vision sensors to localize themselves within their environment. The former is necessary in unmapped or dynamic environments to ensure the safety of the robot and the safety of its surroundings. The latter is necessary because, over time, errors accumulate in the robot's internal location sensors. This paper first looks at several different approaches to sonar-based obstacle avoidance and then examines several different approaches to using vision sensing to localize.

2 OBSTACLE AVOIDANCE

Obstacle avoidance for mobile robots is a topic of much current research. In recent years, two sophisticated obstacle avoidance methods have been implemented: Vector Field Histogram (VFH) [3] and Navigational Templates (NaTS) [20]. These two methods take very different approaches to representing the obstacles in the mobile robot's environment. VFH is the more traditional approach and will be examined first.

2.1 Vector Field Histogram

The VFH obstacle avoidance algorithm uses a two–dimensional Cartesian grid, called the Histogram Grid (illustrated in Figure 1), to represent data from ultrasonic range sensors. Each cell in the Histogram Grid holds a certainty value that represents the confidence of the algorithm in the existence of an obstacle at that location. This representation was derived from the certainty grid concept originally presented in [18].

The central idea behind a certainty grid is to fuse sonar readings over time to eliminate errors in individual sonar readings. In a typical implementation, each cell of the grid array would represent say a 10cm by 10cm square of the

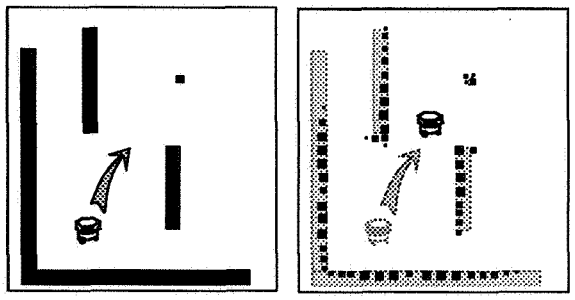

Figure 1: Left figure a robot and an environment. Right figure shows the Histogram Grid of obstacle locations that VFH has created after the robot has moved.

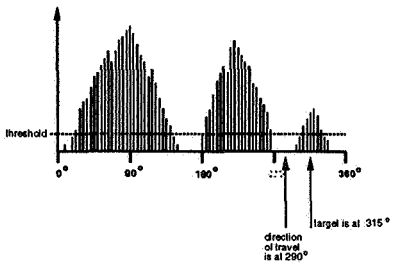

Figure 2: The Polar Histogram has "mountains" in the direction of obstacles. VFH steers the robot toward a "valley" in the direction of the target.

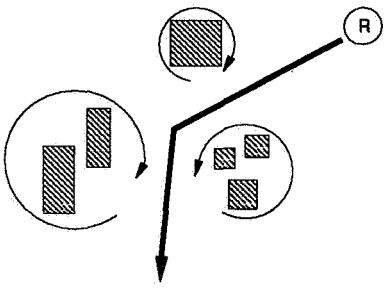

Figure 3: In NaTs each obstacle or group of obstacles is spun. The spin tells the robot to which side of the obstacle it should pass.

environment and the array covers the entire environment space. As the robot travels through the environment, its sonar sensors are continuously being fired and returning range readings to objects. Since the approximate location of the robot at any time is known through odometry and the direction of each sonar sensor is known, the location of objects in the grid array can be estimated. Each time an object is detected at a particular cell location, the value of the cell is increased and the values of all the cells between the robot and this cell are decremented (since they must be empty). The cells have minimum and maximum values, which have been chosen arbitrarily for computational convenience.

To perform the actual obstacle avoidance using the Histogram Grid, VFH creates an intermediate data representation called the Polar Histogram. The purpose of the Polar Histogram is to reduce the amount of data that needs to be handled for real-time analysis while at the same time retaining the statistical information of the Histogram Grid, which compensates for the inaccuracies of the ultrasonic sensors. In this way, the VFH algorithm produces a sufficiently detailed spatial representation of the robot's environment for travel among densely cluttered obstacles, without compromising the system's real-time performance. The spatial representation in the Polar Histogram can be visualized as a mountainous panorama around the robot, where the height and size of the peaks represent the proximity of obstacles, and the valleys represent possible travel directions (see Figure 2). The VFH algorithm steers the robot in the direction of one of the valleys, based on the direction of the target location.

When VFH is combined with an ultrasonic filtering routine called EERUF (Error Eliminating Rapid Ultrasonic Firing) [2] it is uniquely suited to high speed obstacle avoidance. Experiments on a robot at the University of Michigan have demonstrated obstacle avoidance in the most difficult obstacle courses at speeds of up to 1.0 m/sec

2.2 Navigational Templates

One problem with VFH and other obstacle avoidance methods is that it can be difficult for high-level processes to influence low-level behaviors. For example, there is no way to tell VFH to which side of an obstacle it should

pass. Navigational Templates (NaTs) are a method for combining high-level, qualitative guidance with low-level, quantitative control. NaTs come in two forms: substrate NaTs (s-NaTs) and modifier NaTs (m-NaTs). An s-NaT defines a gradient field indicating for each position in space the best direction of travel to accomplish the navigation task. This substrate is independent of any obstacles in the environment. m-NaTs are used to model obstacles in the environment. An obstacle or group of obstacles is represented by an m-NaT that has a spin, which constrains the robot's motion. The spin can be clockwise or counter-clockwise. The robot uses the s-NaTs to determine the general direction of travel and uses the m-NaTs to determine how to avoid obstacles. Figure 3 shows an environment with several m-NaTs and the robots path through the environment.
NaTs has been implemented on several mobile robots. Experiments demonstrate effective obstacle avoidance at top speeds of about 0.2 m/sec. Most importantly, NaTs allow higher-level processes to influence the spin of obstacles and determine the robot's response when it encounters those obstacles. For example, when passing people in a hallway, the robot could always pass to the right of person, even if this wasn't the optimal strategy. This is not possible with traditional obstacle avoidance techniques, such as VFH.

3 LOCALIZATION

All of the obstacle avoidance algorithms discussed in the previous section rely on the robot knowing where it is in the environment, so that it can calculate a travel path to its goal location. Robots keep track of their position and orientation (or their *pose*) in their environment by using wheel encoders. However, due to wheel slippage, etc. the robot's internal calculations of its location will accumulate errors. Over time these errors can become significant. Therefore, the robot needs a way of determining its pose in the environment using exterior references, which is called *localization*. This section will look at three different approaches to localization. The first approach uses CAD-like models of the environment, the second uses triangulation to fixed landmarks and the third uses naturally occurring visual cues.

3.1 Using CAD-like models

A popular and successful way of determining position and orientation is to match visual features to a 3-D, CAD-like world model that has been given to the robot. There are several different, working instantiations of this approach including Harvey [8], Mobi [13], FINALE [12] and COSIM [19]. The basic approach of each system is the same: 1) The robot has a detailed, 3-D model of its environment; 2) A robot action is performed and the model is used to predict the location of visual features that the robot should see after performing the action; 3) The locations of these visual features is compared with corresponding features in an actual image taken by the robot and the robot position is then updated. The major difference between systems is in the visual features that they use. Harvey uses vertices, Mobi uses vertical edges that fall within a small band across the center of the image, FINALE uses horizontal and vertical edges and CoSiM uses horizontal edges. Because many of these systems are similar and in the interest of brevity, only one will be examined in detail–COSIM.

3.1.1 COSIM

The COSIM system consists of a mobile robot with a single black and white camera and a 3D model of the robot's environment. COSIM determines its position and orientation by comparing a simulated image generated from the 3D model with an actual camera image. This process consists of five components:

1. Camera calibration routines that use the model and images to determine effective focal length, and the x and y pixel dimensions.
2. A fast and crude orientation corrector that uses vanishing point analysis.
3. An algorithm to match 2D image points to 3D model features.
4. A reverse projection algorithm to produce two sets of 3D match points from a 3D model and 2D image points.
5. An algorithm for determining a registration vector that gives the rotation and translation necessary to align the two sets of 3D points. This registration vector can be used to determine the camera's position and orientation relative to some fixed coordinate frame that is inherent to the 3D model.

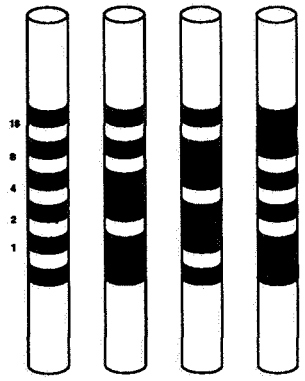

Figure 4: Example object tags showing the basic pattern and objects with bit pattern of 0, 5, 10 and 17.

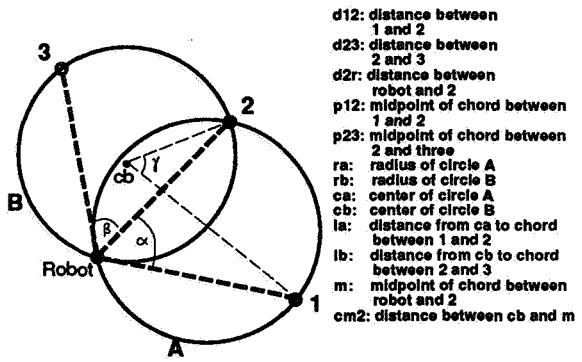

Figure 5: CARMEL's three object triangulation using the circles method.

These five components work together as follows: First, camera calibration (1) is performed to obtain the camera's intrinsic values, which are crucial in all of the other calculations. Calibration will have to be repeated anytime the camera is replaced or when the optical system is adjusted. Next the robot starts moving through the modeled environment and takes images. Each image is analyzed to determine the deviation between the position of the vanishing point in the image from its expected position based on the model and dead reckoning information. This step (2) assumes that there exists a pair or more of dominant parallel lines in the environment (the baseboard lines between the walls and the floor are used in the actual implementation). This technique corrects gross orientation errors very quickly, which allows for expectation matching of features to be performed robustly. COSIM matches image features to model features using a technique based on searching small regions in the image space for particular oriented features (lines) and their intersections. Then a global consistency check is performed in 2D space that eliminates gross matching errors. This produces a set of image points and a matching set of model points. These points are then reverse projected to 3D space. Finally, using a technique based on [1] the two sets of matched 3D points are used to find a correction vector $\vec{q}$. The correction vector is made up of two vectors, a unit rotational quaternion and a translation vector. The correction vector can be used to align the robot with the model, both in position and orientation.

3.2 Using fixed landmarks

Model-based system require a complete model of the environment. This can expensive to produce and maintain. Triangulating to fixed landmarks in the environment can provide position and orientation information without the overhead of a 3-D model. Landmark-based systems rely on a small set of distinctive landmarks that can be seen over a large portion of the environment. Triangulation requires distinct, point-like landmarks that can be recognized from any direction and from long distances. These landmarks can be natural (see [16, 22] for a system that use natural landmarks) or artificial. There are numerous references for triangulation, including [21, 14]. The triangulation approach will be demonstrated in this subsection by examining a robot system, CARMEL, which uses artificial landmarks whose locations have been acquired previously by the robot. First, the landmark recognition system will be described and then the landmark triangulation algorithm will be given.

3.2.1 Landmarks

CARMEL's vision system consists of an object tag design and an algorithm for detecting and distinguishing those tags. The object tag consists of a black and white stripe pattern. An example object tag is shown in Figure 4. The basic stripe pattern is six evenly spaced horizontal black bands of 50 mm width, with the top of the top band and the bottom of the bottom band spaced 1000 mm apart. The white gaps between the black bands correspond to the bit positions in a five-bit word. A white space between two bands corresponds to an "off" bit, while filling the space with black corresponds to an "on" bit. The five bits between the six bands can then represent 32 unique objects. The actual algorithm for extracting objects from an image requires no preprocessing of the image. The algorithm

makes a single pass over the image, going down each column of the image looking for a white-to-black transition that would mark the start of a potential object. A finite state machine keeps track of the number and spacing of the bands. After finding enough bands to comprise a tag the algorithm stores the tag id and pixel length. Once a column is complete, the eligible objects are heuristically merged with objects found in previous columns. The distance between the top of the top band and the bottom of the bottom band, in terms of the number of pixels in the image, is then used to estimate the actual distance from the camera to the object. The algorithm has an effective range of about 12 meters. See [9] for a detailed description and analysis of the vision algorithm.

3.2.2 Triangulation

The landmark recognition algorithm returns a heading and a distance to any landmark. The triangulation algorithm requires only the relative heading between any three known landmarks to determine the robot's location and orientation. Referring to Figure 5, landmark 1, landmark 2, and the robot form one circle (circle A). Even though the robot's location is not known, there are only two possible circles because the angle between landmarks as viewed by the robot is known. Landmark 2, landmark 3, and the robot also form another unique circle (B). From the information available (landmark locations and the angles α and β) the equations of the circles can be determined. The two circles intersect at landmark 2 and the robot's location. Landmark 2's location is already known and the robot's location is the other intersection point. Below is the algorithm for circle intersection. Refer to Figure 5 for variable definitions used. A landmark's position is denoted by (Lxi, Lyi) where i represents the number of the landmark. A landmark's orientation from the robot is denoted by Loi.

1. Properly order landmarks, see below.
2. $\alpha = \text{Lo2} - \text{Lo1}$. if α is too small or equals 90° or 270°, return with error because division by 0 will occur.
3. $\beta = \text{Lo3} - \text{Lo2}$, if β is too small or equals 90° or 270°, return with error because division by 0 will occur.
4. $ra = \frac{d12}{2\sin(\alpha)}$
5. $rb = \frac{d23}{2\sin(\beta)}$
6. $la = \frac{d12}{2\tan(\alpha)}$
7. $lb = \frac{d12}{2\tan(\beta)}$
8. Let $v12x$ and $v12y$ be the unit vector from landmark 1 to landmark 2.
9. Let $v23x$ and $v23y$ be the unit vector from landmark 2 to landmark 3.
10. $cax = p12x - la(v12y)$
11. $cay = p12y + la(v12x)$
12. $cbx = p23x - lb(v23y)$
13. $cby = p23y + lb(v23x)$
14. Return an error if the centers of the two circles are too close (we used 10 units).
15. If γ is very large, then return error.
16. Let $cbax$ and $cbay$ be the unit vector from the center of circle B to the center of circle A.
17. $d2r = 2rb\sin(\gamma)$
18. $c2m = rb\cos(\gamma)$
19. Robot x position $= Rx = 2mx - Lx2 + 0.5$

20. Robot y position = $Ry = 2my - Ly2 + 0.5$

21. $\phi = \arctan(\frac{Ly1-Ry}{Lx1-Rx})$, the heading of landmark 1 from the true robot position.

22. If $\phi > 0.0$ then robot orientation error $= -(Lo1 - \phi)$ else robot orientation error $360° + \phi - Lo1$

23. Return with solution

For this algorithm to work properly, both α and β must be less than 180°. When this condition holds we say that the landmarks are "ordered" properly. Properly ordered landmarks assure that the desired solution (out of the two solutions possible) is found. Properly ordered landmarks have the following two features:

1. They are labeled consecutively (1,2,3) in a counterclockwise fashion,
2. The angle between landmarks 1 and 2 (β) and the angle between landmarks 1 and 3 (α) must be less than 180°.

This triangulation algorithm is analyzed and compared with other triangulation algorithms in [5].

3.3 Using visual cues

A drawback to landmark triangulation is that it relies on knowing the exact location of fixed landmarks and on being able to see them from anywhere in the robot's environment. Such assumptions make these algorithms difficult to use in environments such as office buildings, where the field of view is very restricted. In these environments it may not be possible to localize at any place, instead, the robot may have to do *place recognition*. Place recognition involves using naturally occurring visual cues to determine whether or not the robot is at one of a number of predefined locations in the environment. For example, in an office environment, the robot could use visual cues to determine in which room it is located. However, when the robot is traveling between rooms it does not know its precise location in the environment. Representations that store only interesting places and information about how to travel between interesting places are called *topological maps* and were pioneered by Brooks [4] and Kuipers [15]. Topological maps have become quite popular in mobile robotics and there are many implementations, including [6] and [17]. This section will focus on one particular system, RPLAN [10].

RPLAN consists of two modules, a sonar sensing module that detects interesting places in an indoor, office-like environment and a vision module that identifies those interesting places. Interesting places are openings that the robot finds as it follows a hallway or follows a wall. These openings presumably lead to new spaces that the robot can explore. At each of these openings, the robot takes an image. Vertical edges are extracted from a black-and-white image by a modified Sobel edge detector (see Figure 6). A second image is analyzed in the same way as the first, but it is offset by 18cm from the first image. Once there is a list of edges from this pair of images the edges are matched across the image using the direction of transition of the edge (i.e., was the edge from light to dark or dark to light?), length, and location. The pixel shift in the matched edges from the first image to the second image (called the disparity) is calculated and used to determine a rough distance to the edge. Each visual cue, thus, has three scene-independent features: direction, length, and distance. These cues are stored and then, when the robot comes back to the same place, it can match the cues that it sees with those that it has stored and determine its position. Experiments in natural, unaltered environments, show that by using a combination of sonar and vision information, place recognition can be extremely robust.

4 AN INTEGRATED SYSTEM

Obstacle avoidance and vision sensing are not issues that should be studied separately, as they are both necessary for an intelligent mobile robot. In this section, I look at a completely implemented system that performs a task using both sonar-based obstacle avoidance and vision-based sensing. The robot is CARMEL, a Cybermotion K2A base with a ring of 24 sonar sensors and a rotating camera (see Figure 7). CARMEL has all of its processing entirely on-board in the form of three PCs. CARMEL's task is to explore a 20m by 20m arena, locate ten objects and approach each object. This task was part of the AAAI'92 robot competition, which CARMEL won [7].

CARMEL's strategy was to mark the 10 objects with the barcode tag described in subsection 3.2.1 of this paper. CARMEL then moved to predetermined points in the arena and performed a visual sweep for objects, noting

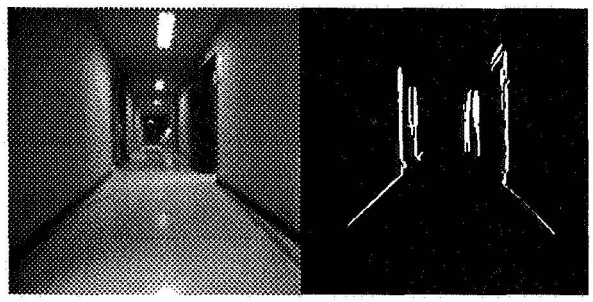

Figure 6: An actual image and its visual cues.

Figure 7: The mobile robot CARMEL.

their locations. When all of the objects were mapped, CARMEL then proceeded to approach each one, choosing the object nearest its current location as the goal. While moving between objects, CARMEL avoided randomly placed obstacles using the VFH obstacle avoidance algorithm described in subsection 2.1. For the competition run, CARMEL's maximum speed was set at 0.5 m/sec, approximately twice the speed of the other robots. CARMEL also had various high-level error recovery routines that allowed it to cope with unexpected situations, such as blocked paths or objects that had been incorrectly located. CARMEL could also triangulate its position using the algorithm described in subsection 3.2 of this paper. However, due to some last minute code changes, the triangulation routines were never invoked during the competition run.During the competition, CARMEL found and visited all 10 objects in just under 10 minutes, about twice as fast as any other robot. Details of CARMEL's software design can be found in [11].

5 CONCLUSION

Perception is an important issue in mobile robot research. If a mobile robot cannot perceive its environment correctly and efficiently, then it will not be able to perform even simple tasks. Sonar sensors and vision sensors are often used for mobile robot perception and they are each suitable for different things. Sonar sensors are typically used to avoid collisions, whereas vision sensors are typically used to perform localization. Robot designers should carefully chose the correct sensor for their task. A successful robot system, such CARMEL, can be developed by carefully integrating sonar and vision sensing.

6 ACKNOWLEDGMENTS

Ulrich Rashke and Dr. Johann Borenstein reviewed the VFH sections of this paper. Charles Cohen and Frank Koss reviewed the triangulation section of this paper. Clare Congdon drew several of the figures. The CARMEL robot is part of the University of Michigan Artificial Intelligence Laboratory. Funding for its purchase and for research using it is provided by Department of Energy grant number DE-FG02-86NE37969.

REFERENCES

[1] Paul J. Besl and Neil D. Mckay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 1992.

[2] Johann Borenstein and Yoram Koren. Noise rejection for ultrasonic sensors in mobile robot applications. In *The Proceedings of the IEEE Conference on Robotics and Automation*, 1992.

[3] Johann Borenstein and Yorem Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), 1989.

[4] Rodney A. Brooks. Visual map making for a mobile robot. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.

[5] Charles Cohen and Frank Koss. A comprehensive study of three-object triangulation. In William J. Wolfe and Wendell H. Chun, editors, *Mobile Robots VII.* SPIE, Bellingham, Washington, 1993.

[6] Thomas Dean, Kenneth Basye, Robert Chekaluk, Seungseok Hyun, Moises Lejter, and Margaret Randazza. Coping with uncertainty in a control system for navigation and exploration. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1990.

[7] Thomas Dean and R. Peter Bonasso. 1992 AAAI robot exhibition and competition. *AI Magazine*, 14(1), 1993.

[8] Claude Fennema and Allen R. Hanson. Experiments in autonomous navigation. In *Proceedings Image Understanding Workshop*, 1990.

[9] Marcus J. Huber, Clint Bidlack, Kevin Mangis, David Kortenkamp, L. Douglas Baker, Annie Wu, and Terry Weymouth. Computer vision for CARMEL. In William J. Wolfe and Wendell H. Chun, editors, *Mobile Robots VII.* SPIE, Bellingham, Washington, 1993.

[10] David Kortenkamp. *Cognitive maps for mobile robots: A representation for mapping and navigation.* PhD thesis, The University of Michigan, 1993.

[11] David Kortenkamp, Marcus Huber, Charles Cohen, Ulrich Raschke, Clint Bidlack, Clare Bates Congdon, Frank Koss, and Terry Weymouth. Integrated mobile robot design: Winning the AAAI-92 robot competition. *IEEE Expert*, 8(4), August 1993.

[12] A. Kosaka and Avi C. Kak. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *Computer Vision, Graphics, and Image Processing*, 56(2), 1992.

[13] David J. Kriegman, Ernst Triendl, and Thomas O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transactions on Robotics and Automation*, 5(6), 1989.

[14] E. Krotkov. Mobile robot localization using a single image. In *Proceedings IEEE Conference on Robotics and Automation*, 1989.

[15] Benjamin J. Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8, 1991.

[16] Tod S. Levitt and Daryl T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44(3), 1990.

[17] Maja K. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3), 1992.

[18] Hans P. Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.

[19] Yuval Roth, Annie S. Wu, Remzi H. Arpaci, Terry Weymouth, and Ramesh Jain. Model-driven pose correction. In *Proceedings IEEE International Conference on Robotics and Automation*, 1992.

[20] Marc G. Slack. Navigation templates: Mediating qualitative guidance and quantitative control in mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2), 1993.

[21] K. Sugihara. Some location problems for robot navigation using a single camera. *Computer Vision, Graphics, and Image Processing*, 42(3), 1988.

[22] Saburo Tsuji and Shigang Li. Memorizing and representing route scenes. In Jean-Arcady Meyer, Herbert L. Roitblat, and Stewart W. Wilson, editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.* MIT Press, Cambridge, MA, 1993.