Wayfarer: An Autonomous Navigation Payload for the PackBot

Brian Yamauchi

iRobot Research Group, iRobot Corporation, 63 South Avenue, Burlington, MA 01803-4903

Phone: (781) 418-3291, Fax: (781) 345-0201

Email: yamauchi@irobot.com, Web: irobot.com, robotfrontier.com

ABSTRACT

iRobot Research is developing the Wayfarer navigation payload to provide autonomous urban reconnaissance capabilities for the man-portable PackBot UGV. This paper describes the perimeter reconnaissance capabilities of the next-generation PackBot Wayfarer. These capabilities include wall-tracking using a Hough transform applied to LADAR data, perimeter following with obstacle avoidance, automatic flipper deployment for obstacle negotiation, cul-de-sac avoidance, stasis detection and escape, and the automatic generation of occupancy grid maps. We present results from experiments where Wayfarer was able to successfully reconnoiter and map building perimeters in an urban environment. Our goal is to have a fully-operational PackBot Wayfarer prototype by September 2005 and to transition this technology into the PackBot product line, delivering battlefield-ready Wayfarer UGVs in the 2-3 year time frame.

1. Introduction

At iRobot Research, we are developing the next generation of battlefield robots. For the Wayfarer Project, funded by the U.S. Army Tank-automotive and Armaments Research, Development, and Engineering Center (TARDEC), we are developing a modular navigation payload that will provide urban navigation capabilities for the PackBot or any other robot using

the PackBot payload interface. The PackBot Wayfarer will be capable of performing fully autonomous reconnaissance missions in urban terrain.

The Wayfarer Project is an applied research effort that is tightly focused on developing robust techniques for autonomous reconnaissance in urban environments. Instead of attempting to solve the general navigation problem, we are developing specific reconnaissance behaviors that will be useful to Army warfighters in the near term.

Our work is unique in taking a narrow, but deep, focus on the urban reconnaissance task. Instead of developing research-oriented navigation systems that have general capabilities, but limited reliability, we are developing a prototype UGV that demonstrates robust urban reconnaissance capabilities in realistic environments. Following successful completion of this project, our goal is to transition this technology into field-deployable UGVs as part of the iRobot PackBot product line.

A previous article [Yamauchi 05] described the Wayfarer Scaled Vector Field Histogram (SVFH) obstacle avoidance system in detail, so this paper will focus on Wayfarer's perimeter reconnaissance and mapping capabilities.

2. Wayfarer Background

The PackBot Wayfarer is equipped with a SICK LD OEM 360-degree planar LADAR and a Point Grey Bumblebee stereo vision system. The PackBot's organic 700 MHz Pentium III CPU communicates with the SICK LADAR over a 38.4 Kbps RS-232 serial interface. The Wayfarer Payload also includes a Plug-N-Run 700 MHz Pentium III processor that is dedicated to stereo vision processing. The Plug-N-Run captures images from the Bumblebee camera over a Firewire interface and communicates with the PackBot CPU via an Ethernet interface.

The Wayfarer mapping system runs on the robot CPU and receives laser range data and robot position data. As the robot moves through the environment, the mapper uses the range and position data to construct an occupancy grid [Moravec & Elfes 85] map of the world. The occupancy grid is a Cartesian grid in the world coordinate frame, divided into cells (0.2 m x 0.2 m in the current representation), with each cell storing the probability that the corresponding location in space is occupied. All cells are initialized to a 0.5 prior occupancy probability. The mapper periodically transmits a compressed version of the map region surrounding the robot via wireless Ethernet to the OCU display.

3. Basic Perimeter Following Behavior

We use a real-time Hough transform to find the lines in the range data that correspond to building walls. The Hough transform [Ballard & Brown 82] is a computer vision technique that works by transforming image point coordinates into votes in the parameter space of possible lines. Each point corresponds to a vote for all of the lines that pass through that point. By finding the strongest points in the parameter space, the Hough transform can determine the parameterized equations for the strongest lines in the image.

We have interfaced the Hough transform line detector with a perimeter following behavior. The ***follow-perimeter*** behavior attempts to steer the robot so that it is parallel to the strongest line detected by the Hough transform. To prevent the robot from oscillating between two lines that are approximately the same strength, an accumulator array is used to integrate the strength of line orientations over time. For computational efficiency, all lines with the same orientation vote for the same orientation, regardless of the range from each line to the robot. Orientations are grouped into 5-degree bins for a total of 72 bins.

The value of accumulator bin a_i at time t is given by:

$$a_{i,t} = (1-\lambda)\, a_{i,t-1} + \lambda \sum_{\forall j: i\beta < \theta(H_j) < (i+1)\beta} v(H_j)$$

where $a_{i,t-1}$ is the accumulator bin value at the previous timestep, λ is the decay rate (between 0 and 1), H is the set of lines detected by the Hough transform, H_j is the jth line from this set, $v(Hj)$ is the number of points voting for this line, $\theta(Hj)$ is the orientation of the line, and β is the bin size.

Note that all orientations are in **world coordinates** not robot-relative coordinates. This is important for correct vote accumulation when the robot is turning. The accumulator is providing a running (exponential decay) tally of votes for particular **global orientations** in the world coordinate frame. Some error will accumulate due to odometry slip, but as long as the decay rate is sufficiently high, the contributions from older line hypotheses will decay rapidly enough so that "blurring" due to odometry drift will be minimized.

The *follow-perimeter* behavior generates a desired heading based on the relative orientation and desired range to the tracked wall:

For left wall following:

$$\theta = \theta_w + k(r_w - r_d) \quad \text{if} \quad k(r_w - r_d) < \frac{\pi}{4}$$

$$\theta = \theta_w + \frac{\pi}{4} \quad \text{if} \quad k(r_w - r_d) \geq \frac{\pi}{4}$$

For right wall following:

$$\theta = \theta_w - k(r_w - r_d) \quad \text{if} \quad k(r_w - r_d) < \frac{\pi}{4}$$

$$\theta = \theta_w - \frac{\pi}{4} \quad \text{if} \quad k(r_w - r_d) \geq \frac{\pi}{4}$$

where θ is the behavior's desired heading in radians (relative to the robot's current heading), θ_w is the orientation of the wall in radians (relative to the robot's current heading), r_w is the range to

the wall in meters, r_d is the desired range to the wall in meters, and k is a constant ($\pi/8$ in these experiments).

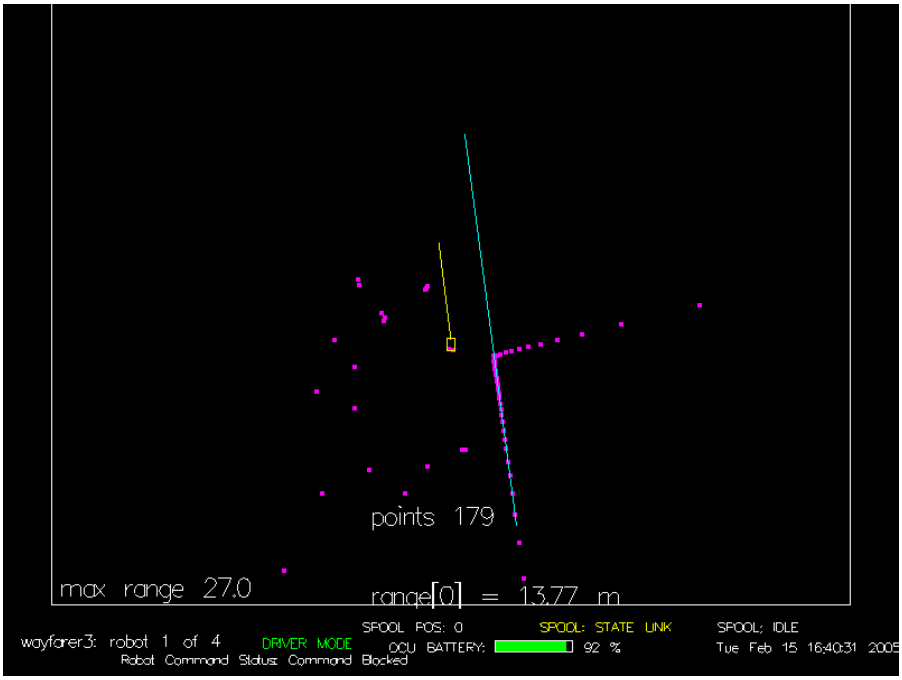

Figure 1: PackBot Wayfarer at building corner

This desired heading is passed to the SVFH obstacle avoidance system. The SVFH then selects the obstacle-free heading that is closest to the desired heading output by *follow-perimeter*. This allows the robot to reactively steer around obstacles that are located next to walls and then resume wall-following automatically.

We tested the perimeter following system in the urban environment outside the iRobot Corporation headquarters in Burlington, MA. Our experiments show that the Hough transform works well to detect building walls. Figure 1 shows the laser scan from the PackBot Wayfarer as it passes a building corner. The dots indicate individual laser range readings. The blue line indicates the position and orientation of the strongest line detected by the Hough transform, which corresponds to the building wall. The yellow line represents the desired accumulator heading of the perimeter following behavior, which is parallel to the wall. As the robot passes

the corner, this heading rotates to match the adjacent (perpendicular) wall. The scattered range points to the left of the robot are range returns from trees.

4. Advanced Perimeter Following Behaviors

The basic *follow-perimeter* behavior is capable of following building perimeters in most cases. However, there are special cases where this behavior can run into problems. These cases include cul-de-sacs and tight spaces where the robot can get stuck. To deal with these circumstances and make Wayfarer's perimeter reconnaissance capabilities more robust, we have developed three advanced perimeter following behaviors: *automatic flipper deployment*, *cul-de-sac avoidance*, and *stasis detection/escape*.

4.1. Automatic Flipper Deployment

In some cases, the robot can encounter obstacles that are below the plane of the laser but are difficult to detect using the vision system (black asphalt curbs, for example). To assist the robot in climbing over these obstacles, we have developed an automatic flipper deployment behavior. When the robot is attempting to climb an obstacle, but its main tracks are unable to lift the robot over the obstacle, the motor currents will rise. The flipper deployment behavior monitors these currents, and when either the left or right drive motor current exceeds a threshold (15 amps), this behavior extends the flippers forward to assist in climbing the obstacle. The flippers remain deployed for a minimum of 10 seconds. When both drive motor currents drop below a second, lower threshold (2 amps), the flippers are retracted back to their home position.

Figure 2: PackBot Wayfarer automatically deploys flippers to climb over obstacle

4.2. Cul-de-Sac Avoidance

In previous experiments, we found that the robot could occasionally become trapped in cul-de-sacs. The robot would follow a wall into a cul-de-sac, then turn around and start to emerge, but then end up following the same wall back into the cul-de-sac again.

To prevent this, we have developed a cul-de-sac avoidance behavior. This behavior remembers the recent locations of the robot and prevents the robot from getting trapped in a loop. The behavior maintains a trace of the robot's recent positions and treats each point in this trace as an obstacle, which is passed to the SVFH obstacle avoidance system. The robot then steers away from its recent path and moves toward unexplored space instead.

Figure 3 (left) shows the robot navigating into a cul-de-sac. The red points are range readings, and the blue line indicates the strongest line detected by the Hough transform. The blue points show the robot's recent path. We maintain a 20-second long history with the robot's odometry position sampled once every 0.2 seconds for a total of 100 position samples. We selected this history length because it is generally long enough for the robot to steer clear of any cul-de-sac opening in which it might become trapped. This is only a function of the size of the cul-de-sac's mouth and not its length or area. When the robot senses the end of the cul-de-sac, it turns around. By ignoring immediately adjacent path points, the robot is able to turn around and steer across its previous path out of the cul-de-sac.

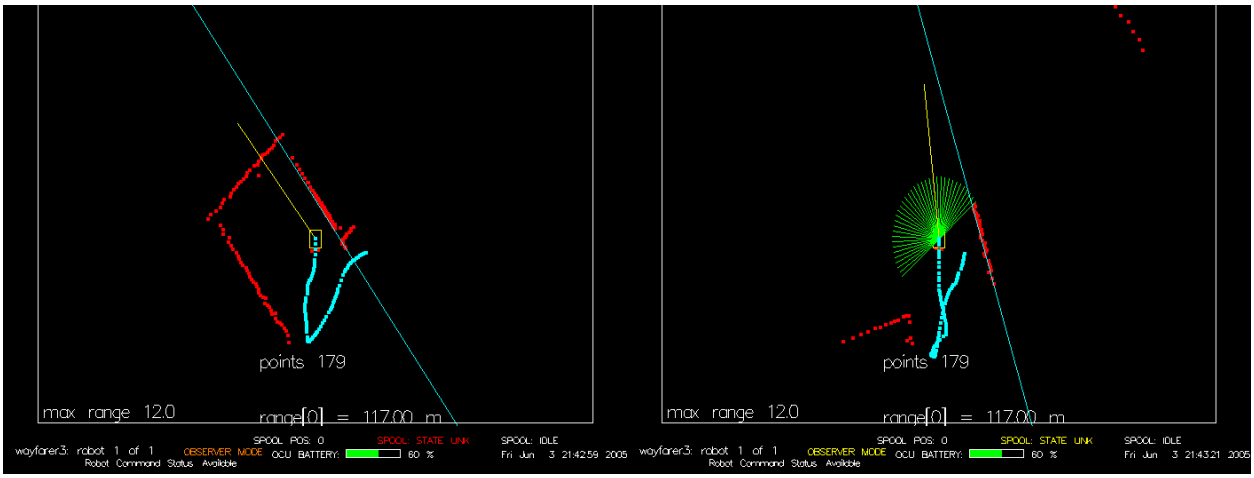

Figure 3: Robot navigates into (left) and out of (right) cul-de-sac

In Figure 3 (right), the robot has left the cul-de-sac. In previous experiments, the robot would sometimes turn around again to avoid obstacles, and follow a wall back into the cul-de-sac. The cul-de-sac avoidance behavior prevents this. Note that even though there are no actual obstacles in the cul-de-sac, the SVFH behavior treats the previous path points as obstacles, and does not indicate any clear (green) paths back into the cul-de-sac.

If the robot were navigating down a very long and narrow cul-de-sac, previous path points could prevent it from initially turning around. In that case, the robot would wait until the

past history expired and then obstacle avoidance would lead it back out of the cul-de-sac. When the robot emerged, the cul-de-sac behavior would prevent it from going back into the cul-de-sac.

We found that adding this behavior drastically increases the robustness of the perimeter reconnaissance system in our urban test environment. Prior to the addition of this behavior, the robot would get trapped in cul-de-sacs in roughly one-third of its perimeter reconnaissance attempts. After the addition of the cul-de-sac avoidance behavior, the robot has successfully escaped 100% of cul-de-sacs encountered.

4.3. Stasis Detection and Escape

Occasionally, when the robot was operating very close to obstacles in cluttered environments, we found that it could get stuck on a low obstacle adjacent to the rear treads. This would occur when the robot attempted to turn and contacted an obstacle that was too low to be detected by the robot's LADAR and vision system, but too high for the tracks to simply slide over sideways during the robot's rotation.

This is an example of the general problem of behavioral stasis, which occurs when the robot is attempting an action, but is "stuck" and unable to move. Back in 1993, we developed techniques for escaping static and cyclic behavior in autonomous agents [Yamauchi & Beer 93]. These were implemented in simulation, but never before applied to real robots.

To increase the general robustness and capability of the Wayfarer system, we implemented a general *stasis-escape* behavior based on our previous work. This behavior detects when the robot is stuck and then attempts random motions until the robot becomes unstuck.

The stasis-escape behavior maintains a stasis level variable. This variable is increased whenever the behavior system is sending a translation or rotation command, but the robot's

treads are not moving (as determined by odometry). The stasis level is reduced whenever the robot is moving. When the stasis level, exceeds a threshold, an escape action is triggered. The escape action commands the robot to move at a random speed (-0.25 to +0.25 m/sec) and a random rotation (-1.0 to +1.0 radians/sec).

If the robot starts moving, the stasis level begins to fall, and when it falls below threshold, the escape action is terminated, and control is returned to the robot's regular behaviors. If the robot does not start moving, then after a specified period of time (2 seconds), another random escape action is generated, and the new translation and rotation commands are sent to the robot. The stasis-escape behavior repeats this process until the robot starts moving.

The general motivation for stasis-escape is that, in an operational mission, it would be far preferable for the robot to risk minor damage then to remain stuck and never deliver valuable intelligence to the soldiers waiting for its return. In practice, the stasis-escape behavior successfully allows the robot to escape from stasis in the current environment, and should significantly increase robustness as we move to more complex environments.

5. Perimeter Reconnaissance Experiments

We now have a fully-operational perimeter reconnaissance system with both autonomous perimeter following and map-building. We have tested this system using an auxiliary building (Figure 4) next to iRobot Headquarters. This building is roughly square and approximately 25 m x 25 m. In these trials, the cruise speed of the robot was set to 0.5 m/sec. The cruise speed is the speed of the robot when not steering to avoid obstacles.

The PackBot Wayfarer is able to fully-autonomously traverse the 100-meter perimeter of this building. The Hough transform is able to reliably detect the locations and orientations of

exterior buildings walls and the obstacle avoidance system can reliably steer the robot around any obstacles in its path.

The robot is started 1-2 meters from any building wall, roughly aligned with the direction of travel, and is commanded to traverse the perimeter maintaining a specified distance from the wall (1.5 meters in these experiments). The robot then automatically follows the contours of the wall, steering around any obstacles encountered – including vehicles, bushes, trees, and other assorted debris. As the robot traverses the perimeter, the mapping system automatically builds a map of the terrain using laser range data.

Figure 4: PackBot Wayfarer performing fully-autonomous perimeter reconnaissance

We compared two different methods of estimating the robot's position for mapping: pure odometry and hybrid compass/odometry. It is important to note that the autonomous perimeter following behavior does *not* rely on any estimate of the robot's absolute position. The reactive *follow-perimeter* behavior operates directly off the Hough transform estimates of the position of nearby walls relative to the robot, without the use of any absolute position information. However, accurate localization is required to build accurate maps of the environment.

For pure odometry localization, track odometry information was used to compute both the robot's orientation and position. For hybrid compass/odometry localization, the compass was

used to determine the robot's orientation, and odometry was used to determine the distance translated between updates. The robot's new position was determined using the following equations:

$$\Delta_t = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}$$
$$x'_t = \Delta_t \cos\theta_t$$
$$y'_t = \Delta_t \sin\theta_t$$

where (x_t, y_t) is the odometry position at time t, θ_t is the compass heading at time t, Δ_t is the distance traversed between time t-1 and time t, and (x'_t, y'_t) is the hybrid compass/odometry position estimate for time t.

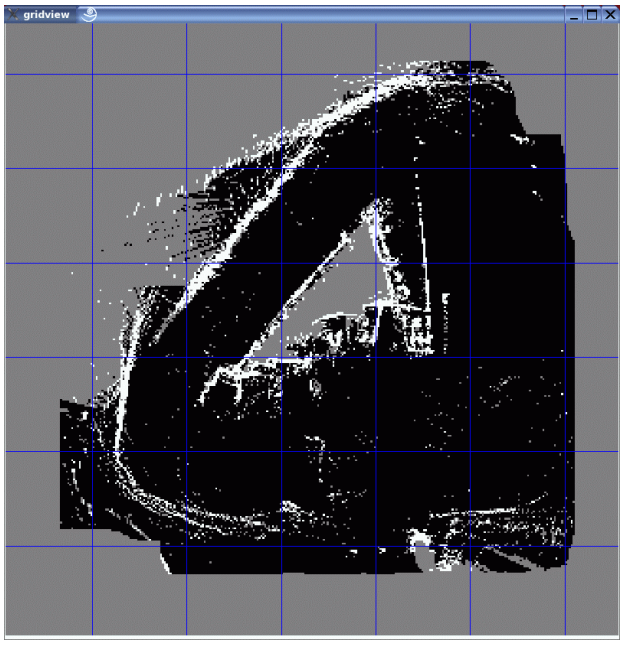

Figure 5: Perimeter map generated using pure odometry

Our experiments showed that pure odometry rapidly accumulated error in the estimate of the robot's orientation. Over a single traverse of the building perimeter, pure odometry accumulated over 90 degrees of orientation error. As shown in Figure 5, this results in the square building outline being distorted into a narrow wedge. In these maps, open space is black,

obstacles are white, and unknown territory (such as the building interior) is gray. The blue lines are spaced at 10-meter intervals.

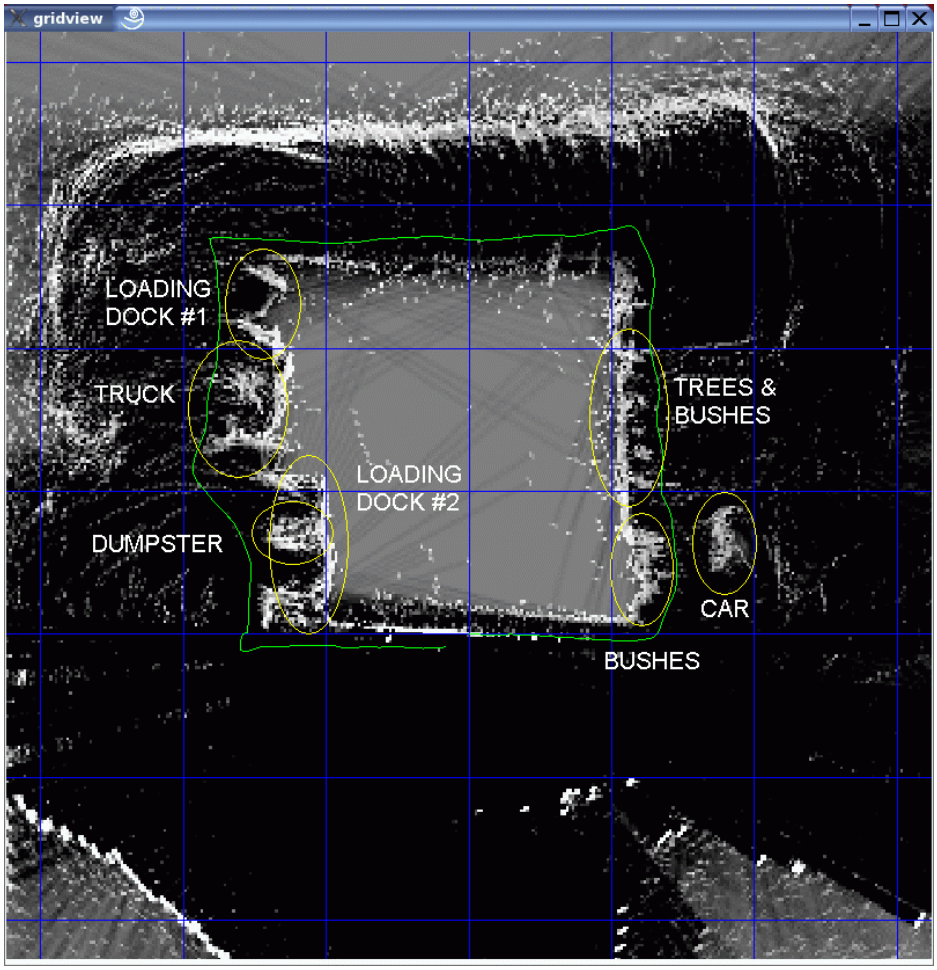

Figure 6: Perimeter map generated using hybrid localization (green line shows robot path, counterclockwise)

In contrast, the compass was able to reliably determine the robot's position to within a few degrees, and the hybrid compass/odometry localization method was able to determine the robot's position accurately to within a few meters throughout the perimeter reconnaissance. Figure 6 shows a perimeter map that was generated automatically by Wayfarer using autonomous perimeter following with hybrid compass/odometry localization. The green line indicates the robot's path, as captured from the hybrid compass/odometry localization system.

The robot was started roughly parallel to the wall, facing east, at the point indicated by the start of the green line south of the building. The robot then followed the building walls, avoiding obstacles, and turning around each corner. The south and north walls of the building were relatively clear of obstacles. However, the east face of the building was fronted with bushes and trees and had a grassy lawn that was raised above the parking lot by a curb. In these trials, the robot was able to steer between the bushes and a parked car (visible in maps), climb up this curb (at the south end of the lawn), follow the line of bushes and trees (visible in the maps), and then climb down the curb (at the north end of the lawn) back onto pavement. The west side of the building included a number of large obstacles, including a parked truck, a large dumpster, and two concrete loading docks (all visible in map). The robot was able to successfully navigate around all of these obstacles and return to its original starting position with an overall position error of less than 3 meters (estimated).

6. Future Work

We have begun initial trials using perimeter reconnaissance to map the exterior of the much larger iRobot Headquarters building. The HQ building is approximately 10 times as long and 6 times as wide as the auxiliary building, so we estimate the perimeter length to be roughly 8 times as long. Initial results have been very promising, though some additional behaviors will be needed to deal with the HQ's more complex terrain. In the future, we will develop a street following behavior and an intersection detection system, both using the Hough transform for detecting building and street orientations. Our plan is to have a fully-autonomous urban reconnaissance capability for the PackBot Wayfarer prototype by September 2005 and to transition this technology to the PackBot product line in the 2-3 year time frame

7. Acknowledgments

This work was funded by the U.S. Army Tank-automotive and Armaments Command (TACOM) Tank-Automotive Research, Development, and Engineering Center (TARDEC) under contract DAAE07-03-L147. Rainer Gasche was the lead mechanical engineer on Wayfarer; Erik Schoenfeld provided mechanical engineering support; and Dan Grollman wrote the Hough transform software.

8. References

[Ballard & Brown 82] Dana Ballard and Christopher Brown, *Computer Vision*. Upper Saddle River, New Jersey: Prentice-Hall, 1982.

[Borenstein & Koren 89] Johann Borenstein and Yoram Koren, "Real-Time Obstacle Avoidance for Fast Mobile Robots." *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 5, September/October 1989, pp. 1179-1187.

[Moravec & Elfes 85] H. Moravec and A. Elfes, "High Resolution Maps From Wide Angle Sonar." In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pp. 116-121.

[Yamauchi 05] Brian Yamauchi, "The Wayfarer Modular Navigation Payload for Intelligent Robot Infrastructure." In *Unmanned Ground Vehicle Technology VII*, edited by Grant Gerhart, Charles Shoemaker, and Douglas W. Gage, Proceedings of SPIE, Volume 5804 (SPIE, Bellingham, WA, 2005), pp. 85-96.

[Yamauchi & Beer 93] Brian Yamauchi and Randall Beer, "Escaping Static and Cyclic Behavior in Autonomous Agents," in *Proceedings of the Second European Conference on Artificial Life (ECAL 93)*, Brussels, Belgium, 1993.