Sensor Based Obstacle Avoidance and Mapping for Fast Mobile Robots†

Chris Gourley and Mohan Trivedi

Computer Vision and Robotics Research Laboratory
Electrical and Computer Engineering Department
The University of Tennessee, Knoxville, TN 37996-2100
gourley@falcon.engr.utk.edu

Abstract

This paper describes one aspect of a project whose goal is to move a robot in an unknown environment and find pipes to decommission. While moving through the environment a low level map, in the form of an occupancy grid, along with detailed location of pipes in the environment are obtained. This paper will deal only with the low level "reflex" of obstacle avoidance that is performed while the robot moves along its path as well as some of the higher level tasks involved in path planning and robot motion in order to negotiate through a hazardous environment. The overall high level interaction with the system is made as simple and user friendly as possible using a graphical interface to control high level tasks. All low level communications and processing are transparent. The robot used for experimentation is a wheeled mobile platform equipped with many different sensors including ultrasonic range sensors and cameras. A quick and efficient obstacle avoidance algorithm has been developed using sixteen ultrasonic range sensor and one infrared proximity sensor.

1 Introduction

Many applications require the ability to navigate a mobile robotic platform through environments containing obstacles, some known, some unknown. For example, many old nuclear plants being decommissioned have no plans or models of the details of what is inside them. Sending humans into such unknown and hazardous environments is both costly and dangerous. Therefore, a robot which can go safely into these areas knowing little or nothing about what it will encounter is needed. Such a system has been developed, a robot utilizing a set of range sensors and a proximity sensor to safely navigating through the environment while mapping the environment with occupancy grids. A pair of cameras are mounted on the robot to perform localization of pipes using stereo vision. Using this, models of the environment are built which can be used for decontamination and decommissioning of the old pipes [12, 13].

The overall system is very powerful being capable of navigating through its environment, creating a coarse map of the environment and detailed positioning of pipes in the environment which can later be used for scanning and cutting of the pipes. This paper will not deal with the integrated system [6], but only with the mobile platform and algorithms used to perform the path planning and obstacle avoidance.

2 Objective

The objective is to create a system with a very simple high level interface for the user. The lower level task are to be transparent. For example, obstacle avoidance is considered to be a low level task, like a human reflex. The user of the system should never have to worry about this reflex, just the high level tasks. To assist in this and for ease in integration onto other platforms the overall project is keep as modular as possible. The modules are as follows:

- User interface,
- Communications,
- Robot motion,
- Sensor reading,
- Obstacle avoidance, and
- Path planning.

The user interface and path planning are the only high level modules with which the user interacts, the rest are lower level tasks that the user can take for granted. Only the robot motion module is manipulator dependent. All others are designed to be ported to other platforms with a minimum of effort.

3 Obstacle Avoidance Algorithm

There are sixteen ultrasonic range sensors arranged symmetrically around the top of the robot. As with most ultrasonic range sensors, they are noisy [9, 11].

† This research is sponsored by a grant awarded by the Office of Technology Development, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

For the occupancy grid mapping this is taken into account by taking multiple readings. For the obstacle avoidance, however, there is not time to take many readings, so it is assumed that one is sufficient.

3.1 Occupancy Grid

Obstacle avoidance and path planning were originally planned to be based around the occupancy grid technique [7, 8]. Decision making as to whether an occupancy cell contains an object can be as simple as a binary function. If the probability of occupancy is greater than one half, the cell is considered occupied; otherwise, it is not. In this project a slightly different approach will be taken. A dead band between a probability of about 0.3 to 0.7 will be used for an unknown state of the cell. Figure *5* incorporates an occupancy grid created in the lab while the robot was roaming.

Using the occupancy grid as a pseudo image, many basic image processing techniques can be used. For this reason the occupancy grid is stored using an image class with many basic image processing techniques available. Taking advantage of these similarities, obstacle avoidance and path planning can be made simpler [10].

3.2 Obstacle Avoidance

Real-time obstacle avoidance presents the problem of navigating around unknown objects in a dynamic environment [4]. The virtual force field (VFF) method described by Borenstein [1, 2] uses an occupancy grid to determine the position of obstacles in the environment. A VFF is then calculated from these obstacles and the robot is "pushed" away from them.

The obstacle avoidance algorithm implemented here is similar to this concept but varies in that it uses only current sensor readings to calculate the VFF instead of using occupancy grids. The algorithm takes into account immediate proximity as to whether to react to an object. With the robot being used for experimentation, the reliability of the positioning is very low (see Section *5)*. Slippage, accumulated error, and taking the robot to different floors makes the use of occupancy grids to determine VFF's difficult. Borenstein [3] has developed specialized hardware and algorithms to reduce error in sensor readings by taking multiple readings. Also, he has accounted for positional error in his occupancy grids. This does take more computational time, and since this module of the project is sharing computer resource with much more complicated modules, the amount of computational time used needed to be as low as possible.

Therefore, a robust, reactive algorithm has been developed for obstacle avoidance which does not rely on occupancy grids. This algorithm allows for a much quicker reaction time to objects which quickly appear in the environment. The current working algorithm is as follows.

1. Initialize system. Set update speed and timeout distances.
2. If obstacle avoidance is on, read sensors; else, end.

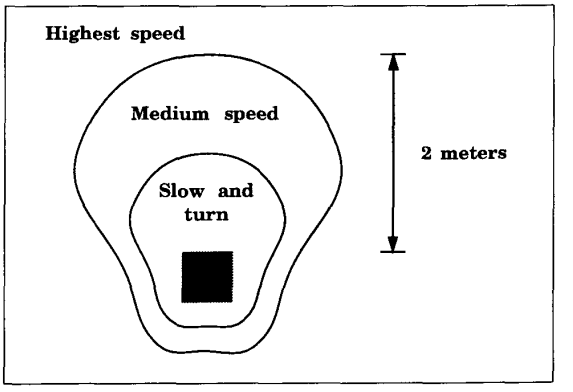

Figure 1: *Thresholds for variable speed and reaction*

3. If reading is under lowest threshold stop, calculate new heading, turn, proceed at lowest speed. Goto *2.*
4. If reading is under second threshold proceed at normal speed. Goto *2.*
5. Proceed at highest speed. Goto *2.*

Each sensor has two thresholds assigned to it creating two rings around the robot (see Figure *1*) to account for impending obstacles. Since the robot moves forward, the front sensors have a higher threshold and the rear have the lowest. The two sets of thresholds allow for variable speed settings while most other mobile platforms performing obstacle avoidance only move at one speed [5]. When none of the sensor have readings less than the highest threshold, no obstacles are near and the robot moves at its highest speed (currently $700mm/sec$). When there is a reading inside the highest threshold the robot slows ($350mm/sec$) to accommodate a possible obstacle. If reading occurs within the lowest threshold the speed is set to its lowest ($150mm/sec$) and a new heading vector is calculated. This heading is based on a scaled inverse of the readings from each sensor (see Figure *2*).

$$\textbf{Heading} = angle(\sum_{i=0}^{12} w_i \mathbf{s}_i) + \textbf{Desired} \qquad (1)$$

where w_i is the weight associated with each sensor and $\mathbf{s}_i$ is the inverse of the vector reading of the sensor. By inverting the reading closer objects apply a large force. In doing this, each sensor gives a weighted vector directed toward the robot. Closer objects give larger vectors. A resultant is calculated and added with a vector in the direction of the motion desired. The angle of this vector determines the new heading. The maximum angle allowed is 45 degrees and the minimum 5 degrees.

This algorithm makes the obstacle avoidance completely reactive in that no previous knowledge of the environment is used. With a sensor update rate of

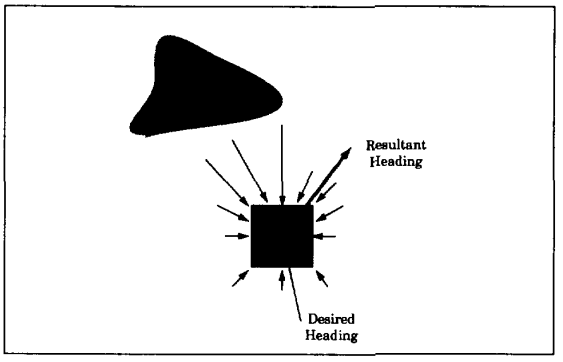

Figure 2: *Sensor vector summation*

four hertz, it is a low level reflex relying on no previous knowledge of the environment. Using this alone the robot can "roam" while creating a map of the environment. Obstacle avoidance is no problem if accurate positioning is not known.

3.3 Path Planning Strategy

Known obstacles are taken into account in path planning. The path planning is based on the original occupancy grid either created earlier or given.

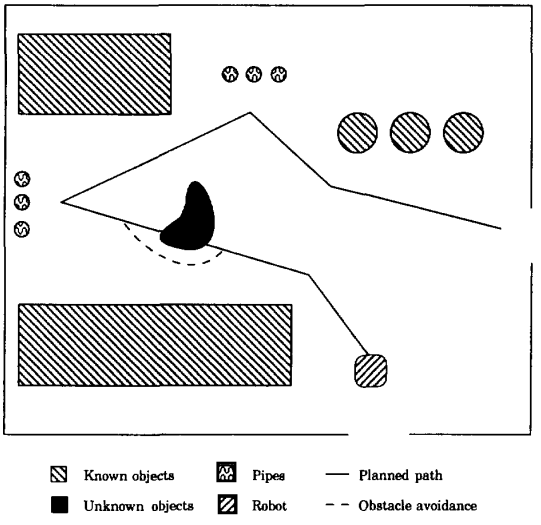

Figure 3: *Results of a typical experiment in path planning*

Path planning is accomplished in three ways. The simplest path planning is to plot a straight line from the start point to the goal. The obstacle avoidance algorithm can then do the rest. As the robot encounters unknown objects in the environment, the obstacle avoidance algorithm will take over. After the robot has steered clear of any danger, it will again try to proceed straight toward the goal.

Another possibility is based on the philosophy of the potential field method. Using the potential method concept to produce the least potential path between to point, simple image processing techniques are used to produce such a path on an occupancy grid. The occupancy grid will be binarized with all cells known to be occupied (probability greater than 0.7) and those which are still questionable (probability between 0.3 and 0.7) will be one level (background or 0) and the cells known and those which are not occupied will be the other (foreground or 1). Since object appear as background and unoccupied space appears as the foreground, the image will first be eroded to grow the occupied areas to allow a margin of area to account for the size of the robot. Two points in the image on the foreground will then be chosen, the robot's current position and the desired position. Now, a thinning or skeletonizing technique is used leaving the two points connected. This will produces a graph from which the smallest path can be chosen. The line connecting the two points should be the least potential path between them. All known obstacles will be avoided. This technique is not yet fully implemented due to positioning errors in the robot, but does prove to be promising.

The last path planning method is based on the philosophy of tele-operation. This method allows the user to manually layout a path for the robot to follow on the user interface.

The path planning really suffers from the lack of positional accuracy. A path that is over ten meters can essentially be rendered useless by accumulation of error. Arriving at a desired point is rare.

4 Implementation and System Configuration

An overall view of the hardware setup is shown in Figure 4. All the hardware used is readily available off-the-shelf items. The mobile platform used for experimentation, known as Experimental Laboratory Vehicle Integrated with Sensors (ELVIS), is a shown in Figure 4. A network board is used to control the robot as well as pass data to and from the sensors. The sensors used are ultrasonic range sensors. A single infrared proximity sensor is also available. Other hardware, including a real time image processing board and cameras, is also available. For this portion of the project, the vision hardware will not be used.

Software development is done on a graphics workstation. The user interface resides on the workstation as well as the high level control for path planning and robot motion. Low level control for the robot and obstacle avoidance is accomplished using a 68030 residing on a VME backplane. The VME rack is currently on board the platform and communication occurs via a wireless ethernet link. Higher level task communicate with the 68030, but the low level task have priority.

4.1 User Interface

The graphical user interface allows high level control of functions in the project, such as path planning, while the low level control remains transparent although still controllable, such as the timeout distance on the sensors. The interface (shown in Fig-

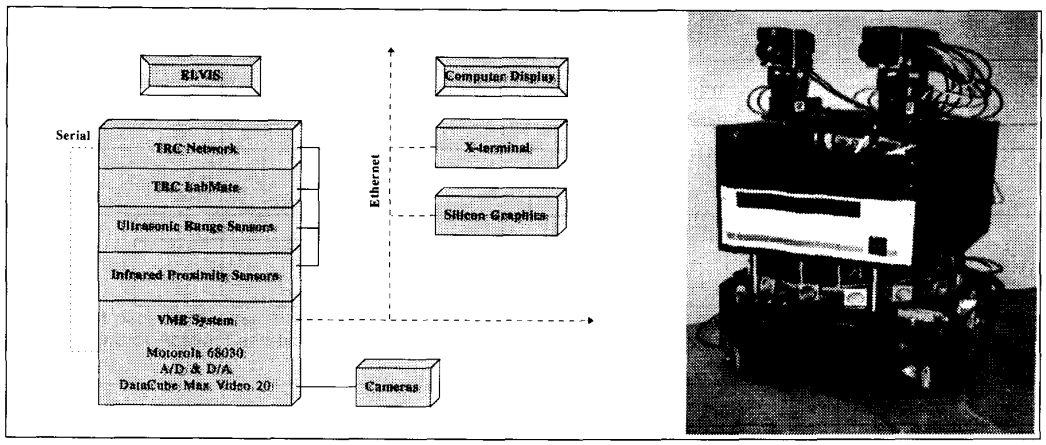

Figure 4: *Hardware configuration for the mobile robotic research*

ure *5*), was created using GL/Motif widgets to run on a graphics workstation and also, without graphics, on any X-terminal. The interface gives the user the ability to change the robot speed, sensor timeout distance, and sensor update speed as well as tele-operated control of the robot. A real-time visual display of the range sensor data is provided, as well as the latest occupancy grid with the position of the robot and the path overlain and the view of the camera on the robot. A software joystick is available for the user to control the motion of the robot when not autonomously moving (see Section *4.3*). An information window supplies the user with the status of the system. Path planning via points can be selected by the user by simply placing points on the map of the occupancy grid (see Section *3.3*). From the user interface the occupancy grid can be saved and previously constructed grids can be loaded.

4.2 Communications

The communications module is user transparent. It is a RPC based client/server program that passes information between the user interface and other higher level routines residing on the workstation and the lower level controls on the VME rack over the ethernet. The server resides on the VME side and receives simple commands from the client to move the robot and read the sensors and encoders returning robot position, heading, and sensor readings. Communication to the network board takes place from the VME through a serial link. The update speed of the RPC calls is set through a user selectable timer. The communications module allows the robot to be controlled from any computer.

4.3 Robot Motion

Positioning of the robot is currently open loop relying on dead reckoning from encoder readings on the wheels. Although programmable, only the speed and direction of the robot are ever changed to keep the algorithms simple and efficient. For this reason only a subset of the possible commands are used. The robot motion occurs in three modes, one tele-operated and two autonomous.

The tele-operated mode uses the software joystick described in Section *4.1*. In this mode, the user simply moves the robot. This is useful for negotiating the robot when it is with in site. The obstacle avoidance algorithm can be used in this mode also so the robot can automatically change headings to avoid objects. The second mode is an autonomous mode used in the path planning. In this mode the robot proceeds to a goal point avoiding obstacles (see Section *3.2*). The last mode is a roaming mode where the robot does not have an ending destination. The robot can roams freely using the obstacle avoidance algorithm. This allows the occupancy grid to be built while the robot is unsupervised or in an unknown environment.

5 Experimentation and Results

Extensive experimentation has been performed using the mobile system. The robot has been run both indoors and out. The algorithm works very robustly. The only problems occurring have been due to the short comings of ultrasonic sensors in general. Some objects which absorb sound well are not seen until the last moment. The update speed of the sensors however allowed the robot to proceed without incident. The only object not seen by the robot was one corner. This problem however is not repeatable and could be a communications problem with the serial link.

5.1 Positioning

A problem with the robot is its lack of positional information. This is well known to be a major problem of which much work has been focused. The manufacturer recommends that dead reckoning using the encoders be limited to about ten meters. After this they suggest using another positioning method such as landmark recognition. Initial testing in the robot encoder error has confirmed this fact. Table *1* shows that the error does increase with increasing distance. This error, however, can be controlled. Tables *2* and *3*,

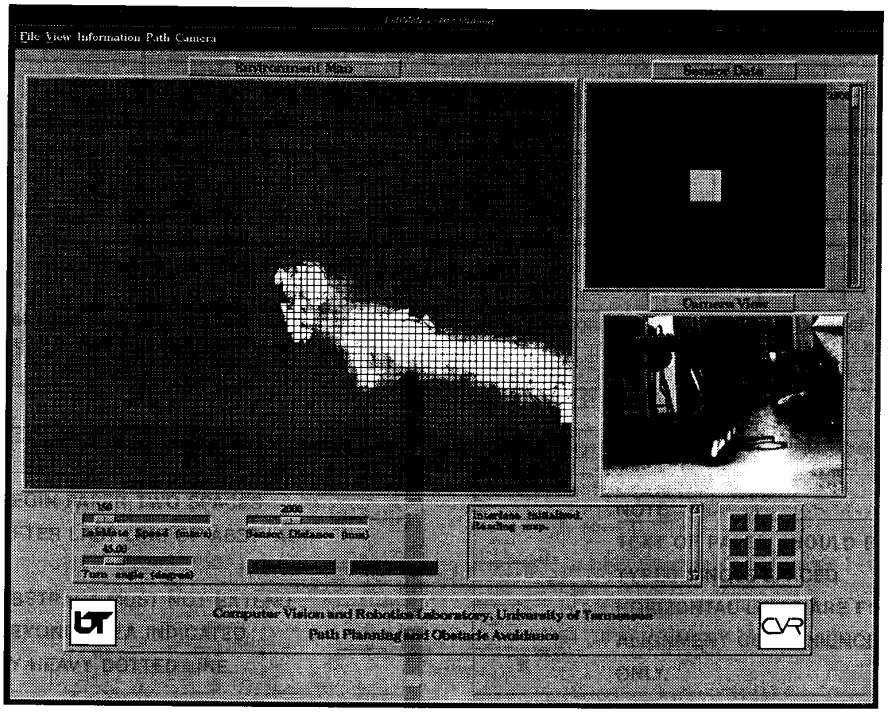

Figure 5: *Graphical user interface for the mobile robot*

Distance (mm)	μ	σ
2000	0.140	0.0865
4000	0.472	0.633

Table 1: *Error measurement in position versus distance traveled*

giving means and standard deviations of errors, ρ, and heading, θ, show that by decreasing velocity and acceleration the error is reduced, mainly to reduction in the amount of slippage induced. This positional

Velocity ($\frac{mm}{s}$)	$\rho(\mu)$	$\rho(\sigma)$	$\theta(\mu)$	$\theta(\sigma)$
100	0.139	0.151	0.341	0.430
200	0.178	0.250	1.453	3.606
250	0.0805	0.542	0.955	4.731
500	0.557	0.263	-1.649	5.342
1000	1.442	2.966	1.044	11.346

Table 2: *Error measurement of position and heading versus velocity*

error leads to problems in calculating VFF's from previously obtained data in that an inaccurate occupancy grid will lead to inaccurate obstacle avoidance. It has also led to problems in the path planning aspect of the project. To compensate acceleration and velocities has to be reduced when more precise occupancy grids are needed.

Accel. ($\frac{mm}{s^2}$)	$\rho(\mu)$	$\rho(\sigma)$	$\theta(\mu)$	$\theta(\sigma)$
100	0.140	0.0865	0.328	0.927
250	0.185	0.331	1.4324	5.434
1000	0.828	2.172	-0.642	8.229

Table 3: *Error measurement of position and heading versus acceleration*

5.2 Results

Even with all the limitations currently imposed on the system, it does a satisfactory job negotiating obstacles both inside and out. The robot has been "herded" onto and off of the elevator with no problem. This definitely causes errors in the two-dimensional occupancy grid since different floors have different obstacles. Overall the system has performed very robustly.

In most areas the robot averages about 0.5 to 0.6 m/s. However, in smaller inclosed areas such as hallways and doors, the robot proceed at an average of 0.2 to 0.3 m/s.

For timing data the robot was run in many different environments. A summary is given in Table *4*. Hallway runs where performed outside the laboratory in a hall about 2 meters wide and 20 meters long. Average speed were fairly low due to the proximity of the walls. During experimentation in the lab, the robot was never more than a few inches from an object,

Distance	Environment	Avg. Speed
20 meters	Hallway	0.322 m/s
100 meters	Outdoors	0.610 m/s
10 meters	Cluttered Lab	0.167 m/s
20 meters	Building Entrance	0.667 m/s
1 meter	Door	0.083 m/s

Table 4: *Average speeds*

thus the lower speed. When approaching doorways, the robot slows and adjusts its heading often further reducing the average speed.

In a more open environment such as outdoors or at the building entrance few objects are around and speed are increased. For the outdoor experiments the robot roamed the sidewalks with two people following to make sure the robot did not get into trouble. In the lab itself the robot was allowed to roam, but the confining space led to low speeds since most of the time was spent turning.

Number of Walls	Environment	Distance (mm.)
1	Hallway	146
2	Laboratory	153
3	Corridor	5

Table 5: *Distance maintained from objects*

To test the absolute limits of the algorithm, the robot, in the roam mode, was sent into a corridor slightly less than a meter wide giving the robot only about 5 inches clearance on each side. The end of the corridor was blocked to see if the robot would be able to return. The robot did return every time, but only about 20% of the time was this accomplished without running into one of the walls at the end of the corridor. For most commonly encountered obstacles, such as walls or corners, the robot has no problem in turning away. From Table *5* about the closest the robot comes is $150mm$.

6 Concluding Remarks

This paper deals with the low level "reflex" of obstacle avoidance that is performed while the robot moves along its path as well as some of the higher level tasks involved in path planning and robot motion in order to negotiate through a hazardous environment. The overall high level interaction with the system is made as simple and user friendly as possible using a graphical interface to control high level tasks and keeping all low level communications and processing transparent. The robot used for experimentation is a wheeled mobile platform and is equipped with many different sensors including ultrasonic range sensors and cameras. A quick and efficient obstacle avoidance algorithm has been created using twelve ultrasonic range sensor and one infrared proximity sensor using weighted sums of sensor readings. This allows the robot to quickly negotiate around obstacles.

References

[1] J. Borenstein and Y. Koren. Real-Time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[2] J. Borenstein and Y. Koren. Real-Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 572–577, Cincinnati, OH, May 1990.

[3] J. Borenstein and Y. Koren. Noise Rejection for Ultrasonic Sensors in Mobile Robot Applications. In *Proceedings of the 1990 IEEE Conference on Robotics and Automation*, pages 1727–1732, Nice, France, May 1992.

[4] R. A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[5] C. Congdon, et. al. Carmel Versus Flakey: A Comparison of Two Winners. *IEEE AI Magazine*, pages 49–57, Winter 1992.

[6] C. Gourley, et. al. An Integrated Mobile Robotic System for Environmental Mapping and Object Localization. In *Proceedings of SPIE Symposium on Optical Engineering in Aerospace Sensing*, Orlando, FL, April 1994.

[7] A. Elfes. Sonar-Based Real-World Mapping and Navigation. *IEEE Journal of Robotics and Automation*, RA-3(3), June 1987.

[8] A. Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *IEEE Computer Magazine*, June 1989.

[9] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotic Control, Sensing, Vision, and Intelligence.* McGraw-Hill, 1987.

[10] R. C. Gonzalez and P. Wintz. *Digital Image Processing.* Addison-Wesley Publishing Company, Reading, MS, 1987.

[11] B. K. P. Horn. *Robot Vision.* The MIT Press, 1989.

[12] S. M. Thayer, et. al. Three-Dimensional Sensing, Graphics and Interactive Control in a Human-Machine System for Decontamination and Decommissioning Accplications. In *Proceedings of SPIE Conference on Intelligent Robotics and Computer Vision*, Boston, MA, November 1992.

[13] S. M. Thayer, et. al. On-Line Stereo Vision and Graphical Interface for Decontamination and Decommissioning Applications Using the Advanced Servo Manipulator. In *Proceedings of the Fifth Topical Meeting on Robotics*, volume 1, pages 287–294, Knoxville, TN, April 1993.