Visually-Guided Obstacle Avoidance in Unstructured Environments

Liana M. Lorigo, Rodney A. Brooks, W. E. L. Grimson
MIT Artificial Intelligence Laboratory
Cambridge MA 02139 USA
Email: liana@ai.mit.edu

Abstract

This paper presents an autonomous vision-based obstacle avoidance system. The system consists of three independent vision modules for obstacle detection, each of which is computationally simple and uses a different criterion for detection purposes. These criteria are based on brightness gradients, RGB (Red, Green, Blue) color, and HSV (Hue, Saturation, Value) color, respectively. Selection of which modules are used to command the robot proceeds exclusively from the outputs of the modules themselves. The system is implemented on a small monocular mobile robot and uses very low resolution images. It has been tested for over 200 hours in diverse environments.

Keywords: Vision-based navigation, space exploration, modular design, reactive control, unstructured terrain.

1 Introduction

This work addresses the problem of developing a mobile robot to avoid obstacles while traveling in unstructured environments, that is, environments for which there is no strong prior knowledge of the appearance of the ground or the locations or appearance of the obstacles. An autonomous vision-based system is presented that performs obstacle detection and avoidance in such environments. It has been implemented on a small mobile robot with a single camera and tested in a range of environments.

The problem of vision-based navigation has been previously examined from a number of different approaches. Variations have included using sensory input from stereo vision, monocular vision, and the combination of vision with other sensors. Methods also vary in how they deal with temporal information, from using individual frames exclusively [6] to computing optical flow fields from multiple frames. Domains include road and off-road travel [5, 10, 14, 3, 16, 4] and indoor robotic navigation [6, 2, 12, 11]. One motivating goal of the current work is autonomous exploration of the surface of Mars. Many researchers are addressing the problem of navigation in this domain [9, 7, 13, 8].

The current system is most closely related to Horswill's work [6], which used low-level environment-dependent algorithms for vision-based navigation. The current work modifies and extends such obstacle avoidance techniques to handle a different class of environments, including domains in which the ground may have rich visual texture. This work also draws on visual routines theory in that it combines separate low-level vision algorithms in a similar manner [15].

Further, the obstacle detection method presented is *reactive*, storing almost no memory of obstacle locations, but rather using current images directly. That is, percepts are converted to actions without the use of complicated internal state. The system also draws from the behavior-based subsumption architecture approach for combining routines [1]. Alternative architectures integrate information from multiple routines according to pre-set weights for each routine [10]. While a part of the current system uses a related approach, the weights are automatically computed instead of pre-set by the user.

Note that the robot's only goal is to move safely within cluttered environments; it has no target location. Applications include exploration, in which video or other data could be acquired, and more general navigation tasks when combined with a technique for moving toward a specific goal location. In a subsumption architecture approach, for example, another layer of behavior incorporating the goal location could be combined with the current obstacle avoidance system.

This paper thus presents a system which deals with unknown environments and obstacles, utilizes a visual routines framework for combining multiple visual cues, and utilizes an environment-dependent algorithms approach to obstacle detection and navigation.

2 Modular Obstacle Avoidance

The high-level structure of the system is illustrated in Figure 1. Each of three visual processing modules takes as input the image frame from the robot's camera and generates a coarse image-based representation

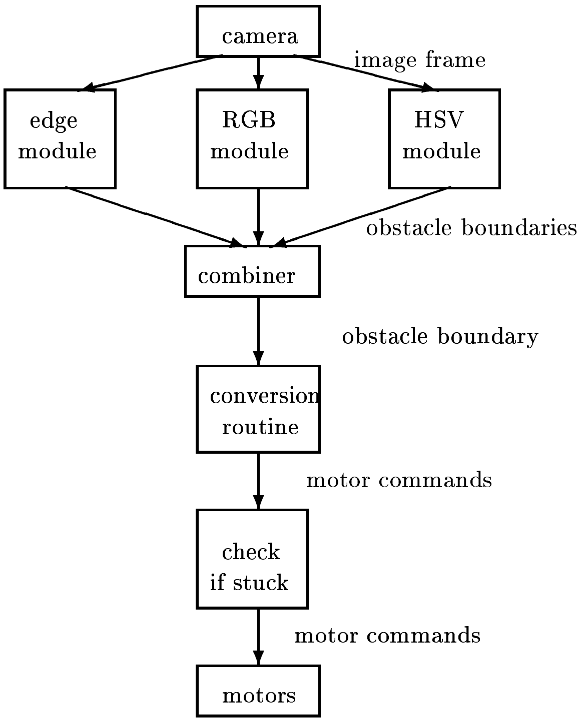

Figure 1: Control flow of system.

of depth in that image. This representation is called an *obstacle boundary*. These obstacle boundaries from the individual modules are combined into a single obstacle boundary which is converted to motor commands. Finally, a routine checks if the robot is "stuck", in which case it is directed to turn in place until a safe path is detected. Else the motor commands from the *combiner* are sent directly to the motors.

2.1 Platform

The system is housed in the Pebbles III Robot, built by IS Robotics, Inc. Pebbles has a tracked base and is driven by independent left and right motors. It is equipped with a Motorola 68332 processor and an uncalibrated Chinon 3mm camera positioned at the front of the robot 10.5 inches off the ground. The system uses the Cheap Vision Machine (CVM) visual-processing hardware system, designed by DIdeas, Inc. and Ian Horswill at the MIT AI Laboratory. The processor is a Texas Instruments C30 DSP. The vision software is written in C and runs on the CVM. The system also runs a control program, written in L (a subset of Common Lisp) on the 68332. All processing is performed on-board the robot, and the images used have a resolution of only 64 x 64 pixels.

2.2 Vision Modules

Each vision module generates an obstacle boundary based on the current image frame. The modules share a common framework, varying only in the particular low-level properties used in the computation of the obstacle boundary. The low-level properties in the current system are color and edge distributions, which are discussed following the discussion of the framework.

Framework. Several constraints underly the framework given below: the ground type at the robot's initial location is considered favorable, boundaries between safe ground and obstacles are visible in single image frames, the ground is flat, and all objects rest on the ground. The last two constraints imply that distant objects appear higher in the image than nearby objects. This property is known as the *ground plane constraint.*

The initial location is favorable, and it is assumed that this favorable ground extend far enough in front of the robot to be visible in the image. This constraint together with the ground plane constraint implies that nearby obstacles can be detected by starting at the bottom of a single image (the bottom region then corresponds to safe ground) and scanning up the image until the type of ground changes. The height of this "change" corresponds to the image height of the obstacle. The measure used to determine this change varies across the modules.

Regarding implementation, the upward scan of the image is performed on vertical slices of the image which are 20 pixels wide, compared with the total image width of 64 pixels. The window that is moved up these slices is 20 pixels wide by 10 pixels high (Figure 2). Vertical slices are taken at horizontal shifts of one-pixel increments, and an obstacle height is obtained for each slice. These heights are stored in a one-dimensional array indexed by the x-coordinate of the (center of the) vertical slice. This array is the *obstacle boundary* for the given image and the given module. Three such arrays are illustrated Figure 4a. In this way, heights are obtained separately at each x-coordinate except for those at the far edges of the image which are never at the middle of a window. There are 10 and 9 such coordinates for the left and right sides respectively since the center pixel is defined to be the 11th pixel of the window. This yields $64 - 10 - 9 = 45$ separate measurements. Moreover, when scanning up the individual slices a module only shifts each window vertically by one pixel to obtain the next window so the height values are found at single-pixel precision.

Module-Specific Measures. The framework just described can be implemented with a variety of measures to yield many different modules. The measures in the current system are each based on the histogram of a different image property over a given window of the image. They compute the value of the module-specific property centered at each pixel in that window and generate a histogram of the values.

Note that each histogram need not be computed

Figure 2: Image from Pebbles' camera, illustrating a vertical slice and a window.

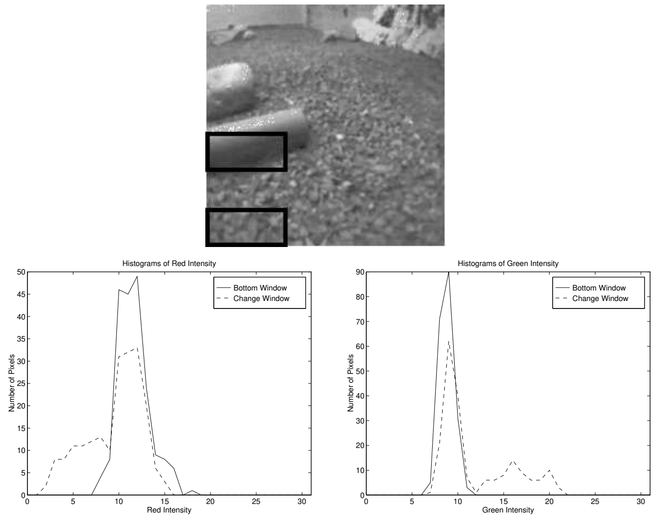

Figure 3: The sample histograms were generated from the windows of the first image. The obstacle seen in the higher window is green, and the ground is gray.

from scratch for each window as the window is shifted up the image. Since the shift occurs in single-pixel increments, the system can simply add the top row of the new window to the histogram and subtract the bottom row of the previous window from the histogram. Horizontal shifts are analogous. This observation significantly reduces computation time.

The metric used for histogram comparison is the area between the histogram of the "safe" (bottom) window and the histogram of the current window. The difference area is given by

$$area = \sum_{v} |h_{current}(v) - h_{safe}(v)|$$

where the v's are the potential values attained by the particular image property, $h_{current}$ is the histogram of the current window, and h_{safe} is the histogram of the bottom window of the current image column. When this area is large, the current window is assumed to contain an obstacle.

The image properties used in the respective modules are brightness gradient magnitude, normalized RGB

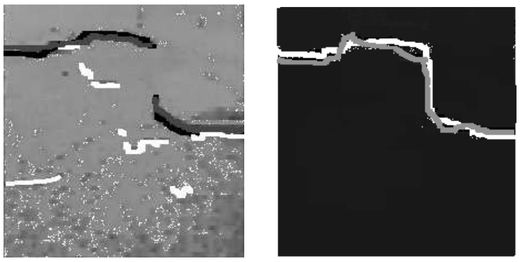

Figure 4: a) Obstacle boundaries from the edge (white), RGB (gray), and HSV (black) modules overlaid on the image. There is a large rock in the upper right portion of the image, correctly detected by the RGB and HSV modules. b) Outputs of the smoothness (gray) and median (white) combination methods, for the individual boundaries shown in (a).

(Red, Green, Blue) color, and normalized HSV (Hue, Saturation, Value) color. Brightness gradient magnitudes are calculated by blurring the image slightly, computing gradients from vertical and horizontal pixel differences, and normalizing the magnitudes to lie in the range [0,31]. To normalize the color modules by intensity, only two dimensions are used instead of three. In the RGB module, only the red and the green channels are used; in the HSV module, only hue and saturation are used. Further, hue and saturation values are ignored for pixels with saturation less than 3.3% of full saturation. The single histogram approach is extended to use a pair of histograms for measures whose value computed at a pixel is actually a pair of values. In such cases, the difference between windows is the sum of the differences in the areas under the corresponding histograms, as illustrated in Figure 3.

2.3 Fusing the Modules

Two methods, one based on smoothness and one based on median, are presented. Only the median method is used by the current system.

Actual obstacle boundaries in the environments considered for this work are likely to be smooth. For example, in Figure 4a, notice that the smooth gray and black arrays are more accurate than the jagged white array. Accordingly, each array is assigned a coefficient proportional to its smoothness, computed as the inverse of second differences. The smoothness coefficient s_i of the array m_i of the ith module is given by

$$s_i = \frac{1}{\frac{1}{n}\sum_{x=1}^{n} |m_i(x+1) - 2m_i(x) + m_i(x-1)|}$$

where n is the width of the image.

An overall obstacle boundary is produced by pointwise averaging the values in the individual arrays, weighted by the smoothness coefficients. The overall array a is defined as

$$a(x) = s_1 * m_1(x) + s_2 * m_2(x) + s_3 * m_3(x)$$

for each x-coordinate.

The second criterion for combining the arrays from the three modules relies on choosing the median value at each x-coordinate. The two methods are illustrated by Figure 4b.

In the final system, the median method is used exclusively; the smoothness method is presented for comparison purposes only. Both the smoothness and median approaches gave very similar results in the test environments, but there are potential situations in which linear combination is inappropriate for sensor fusion. An example is a case of two equally smooth boundaries indicating obstacles at different image locations. The median method was chosen for this reason.

2.4 From Obstacle Boundaries to Motor Commands

The final processing step for each image is the determination of the motor commands for the robot based on the obstacle boundary generated by fusing the outputs of the individual vision modules. The ground plane constraint implies that the lowest obstacle heights in the array of the boundary correspond to the closest obstacles across the image.

Motor Command Computations. The robot should turn away from obstacles that are nearby, and the degree of this turn should depend on the proximity of the obstacle for a smooth trajectory. Thus the desired turn angle is proportional to the difference between the average image heights of obstacles on the left and right sides of the image:

$$turnangle = c_1 * \frac{1}{n/2}(\sum_{x=1}^{n/2} a(x) - \sum_{x=n/2}^{n} a(x))$$

where a is the overall array, c_1 is a constant dependent on the robot's motors, and n is the width of the image.

The forward speed should be large if obstacles are far away and should be small if obstacles are close, perhaps even negative. Accordingly, it is proportional to the average image height of obstacles over the whole length of the image. Some constant k is first subtracted from the average height to achieve reverse motion when obstacles are too close. Varying k changes how close the robot can get to obstacles before backing up. The equation is

$$forwardspeed = c_2 * ((\frac{1}{n}\sum_{x=1}^{n} a(x)) - k)$$

where c_2 is another motor calibration constant.

A simple conversion yields left and right motor commands: $command_{left} = forwardspeed - turnangle$, and $command_{right} = forwardspeed + turnangle$.

One Bit of State. Normally, the commands calculated above are passed directly to the robot's motors. However, when obstacles are very close to the robot, the obstacle boundary appears low in the image, so the corresponding motor commands are small. Thus when these commands fall beneath a threshold, the robot enters the "stuck" state. In this state, the robot is directed to turn in place regardless of the commands computed from the image.

When subsequent commands exceed a larger threshold, indicating a clear path ahead, the robot exits this state and the computed commands resume control of the motors. The bit of information signifying whether or not the robot is in the stuck state is necessary for hysteresis, the use of different thresholds for entering and exiting that state. This bit is the only "map" stored by the system. Otherwise, it is completely reactive.

2.5 Efficiency Analysis

Let w and h be the width and height pixels of the image, m be the width of the window, and n be the length of the histogram. Assuming a fixed number of modules, the running time is $O(wh(m+n))$ per image, as follows. Initial processing such as color conversion and gradient detection is $O(wh)$. For each window location, a row of the window is added to and subtracted from the previous histogram, so generating the histograms is $O(whm)$. Each histogram comparison takes $O(n)$, so that total is $O(whn)$. Combining the modules and generating the motor commands is $O(w)$.

3 Experimental Results

The system has been run cumulatively for over 200 hours. Although more human intervention was required during testing of individual modules and debugging, the level of autonomy of the complete system is demonstrated in section 3.4 below. It has been tested in diverse cluttered environments, including test sites at the MIT AI Laboratory and two simulated Mars sites at the Jet Propulsion Laboratories in Pasadena, California. The results discussed here are those of the complete system, although some discussion of the performance of individual modules is included. The speed of the robot is approximately 0.3 meters/second, and the processing speed is approximately 4 frames/second.

3.1 Test Conditions

Testing sites for the system are pictured in Figure 5. The "sandbox," is a 15ft x 10ft room in which the ground is coarse gravel of various shades of gray, and larger rocks are obstacles. The second site is a lounge area covered with a carpet of varying shades of orange. Obstacles include walls, sofas, bookcases, tables, and chairs. During testing in all locations, people stepped in front of the robot and placed other objects in its path.

The system was tested in two simulated Mars environments at the Jet Propulsion Laboratories (JPL) in Pasadena, California. The first environment was a large room in which the ground was sand and the obstacles were rocks of various sizes, colors, and textures. The second environment was JPL's "MarsYard", an outdoor area approximately 80ft x 60ft in which the ground is reddish brown sand and sloping in places. The colors, textures, sizes, and distributions of rocks are based on those found on Mars.

Success was measured by observing both the robot's behavior and the obstacle boundaries found for individual frames. Successful behavior was defined as traveling around a cluttered environment, moving straight forward when the path ahead is clear and navigating around obstacles when encountered. The obstacle boundaries were transmitted from the robot to a video monitor for verification.

3.2 Individual Modules

All three modules reliably detected obstacles in many scenarios. To contrast the modules, however, several situations are described in which they performed differently.

In the lounge site, both the edge module and the HSV module consistently detected obstacles such as walls, sofas, and people. However, the edge module occasionally missed a smooth brown bookcase, but the HSV module consistently detected it. The opposite situation occurred in the case of a particular style of chairs which had metallic legs that reflected the color of the carpet. In the brightly lit room of sand at JPL, the ground occasionally appeared white in parts of the image due to the camera angle and reflections; HSV is worse than RGB in this situation. Conversely, in the lounge site, the HSV module out-performed the RGB module, which occasionally falsely reported obstacles at areas of the carpet that were slightly faded or in shadow. In the sandbox test site, the color modules were preferable to the edge module due to the variable texture of the gravel. In all five of these cases, the combination of the three modules compensated for the failure of an individual module.

3.3 Complete System

For testing in the sandbox, the obstacles were moved into many configurations, additional obstacles were added, and people interacted with Pebbles by stepping in its way. Normally the space between obstacles was only slightly larger than the robot's width. In this situation, the robot navigated safely for large amounts of time. No effort was made to control the lighting conditions, and, as a result, they often varied. Performance was not harmed by these variations. The run-time of the robot was limited primarily by hardware concerns and occasionally by the failure modes mentioned below.

Figure 5: Testing at various sites: sandbox, lounge, JPL indoor site, JPL outdoor MarsYard

Further, the system avoided obstacles in the lounge. Walls, sofas, boxes, chairs, and people were consistently avoided. Corridor following, even at corners, was easily accomplished by the system. Again, moderate lighting variations caused no difficulty. These results were repeated in other rooms where differences included the type and amount of clutter and the pattern of the carpet. Some pictures of Pebbles operating in various environments are given in Figure 5.

In the JPL indoor room of sand and rocks, the system achieved performance equivalent to that observed in the sandbox test site. It should be stressed that this performance was achieved without any customization whatsoever to this environment, as compared with the initial development environments at MIT. Moreover, performance in the outdoor JPL MarsYard supported the use of this approach on a Mars rover, perhaps in conjunction with other techniques. The system succeeded in many situations in this environment, again with no changes to the code. Difficulty with late afternoon shadows was observed, and is discussed below.

3.4 Example Run

A continuous 20-minute sample run was performed in the sandbox environment whose layout is shown in Figure 6. Figure 7 depicts the approximate trajectory for this run. The X indicates the situation where one of Pebbles' treads became stuck on a rock to the side of the robot, and the robot needed to be moved slightly

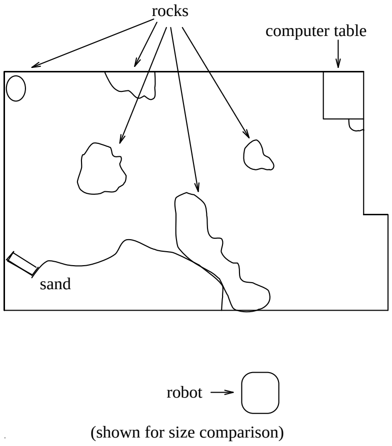

Figure 6: Layout of sandbox test environment.

by hand. The rest of the trajectory was completely autonomous. Many such runs have been performed, and the trajectory displayed is representative of the system's performance. Recall that no environmental map was stored across frames and that the robot had no goal location, but rather motor commands were determined directly from each image.

3.5 Failure Modes

The system failed when obstacles were outside the field-of-view of the camera. This failure mode is not a shortcoming of the vision software, but rather of the hardware configuration of the camera and the robot as compared with the reactive navigation algorithm. That is, when no map is stored, the focal length of the camera must be small enough to fully accommodate the width of the robot. This requirement was not met, so for example, the failure occasionally happened when a small obstacle was directly in front of one of Pebbles' treads, and Pebbles had just turned sharply away from a different obstacle so that the troublesome obstacle had never been within the camera's view.

Regarding the vision algorithms, it is assumed that the floor pattern can be effectively modeled by a single image patch. Consequently, carpets with broad patterns or boundaries between distinct patterns resulted in false alarms. Sharp shadows also posed a problem in bright outdoor sunlight when shadows were sometimes classified as obstacles. Similarly, bright specularities on a shiny floor occasionally caused the system to falsely report obstacles. Prior knowledge of such patterns or an additional method for depth estimation would be required to resolve these issues.

Note, however, that the method presented could be easily extended to use histograms of higher-order statistics of intensities to handle texture changes as well as intensity changes. In this way, it could be applied to a larger class of environments.

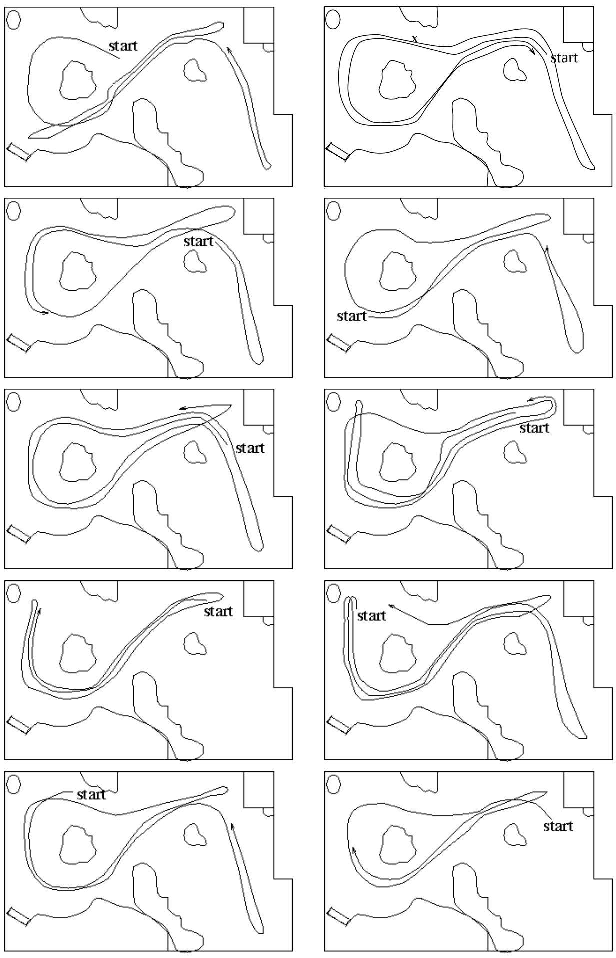

Figure 7: Example: 20-minute continuous run of the system. It is shown across ten (consecutive) images for clarity only.

3.6 Robustness

The system is relatively insensitive to slight changes in thresholds and constants. Thresholds for determining if the measure of a window is different from the stored measure can be varied by about 15 percent without harming performance. Image resolutions and window sizes of up to four times the current dimensions have been tested with good results. Further, performance is not harmed by relaxing the ground plane constraint to included rough, slightly sloping surfaces such as the sandbox test site.

4 Conclusions

This paper describes an autonomous visually-guided obstacle avoidance system implemented onboard a mobile robot using a single uncalibrated camera and very low resolution images. The system is comprised of multiple independent visual processing modules each of which is computationally simple and segments obstacles in the image based on a single visual property such as intensity gradients or color information. Each module receives the image directly from the camera and returns a one-dimensional array indicating the perceived locations of obstacles. These arrays are combined into a single array from which motor commands are generated.

The system is adaptive to new environments since it obtains a new model for ground from any current environment during run-time. Thus, no prior customization or training is needed for many new environments. Further, the described approach of combining multiple computationally simple modules into a system for robustness across widely varying environments can also be applied to multiple cameras, other sensors, or a combination of various sensors.

Acknowledgments

The authors thank Ian Horswill for help with this project. This work was supported by Jet Propulsion Laboratories, Contract 959333, and by a National Science Foundation fellowship.

References

[1] Brooks, R.A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14-23.

[2] Coombs, D. and Roberts, K. 1992. "Bee-bot": using peripheral optical flow to avoid obstacles. In *Intelligent Robots and Computer Vision* Boston, MA, SPIE 1825:714-721.

[3] Crisman, J.D. 1991. Color region tracking for vehicle guidance. *Active Vision*, MIT Press: Cambridge, MA. pp. 107-220.

[4] Dickmanns, E.D., Mysliwetz, B. and Christians, T. 1990. An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles. *IEEE Transactions on Systems, Man, and Cybernetics* 20(6):1273-1284.

[5] Hebert, M., Pomerleau, D., Stentz, A., Thorpe, C. 1995. Computer vision for navigation: the CMU UGV project. In *Proceedings of the Workshop on Vision for Robots*, IEEE Computer Society Press. pp. 97-102.

[6] Horswill, I.D. 1993. Polly: A vision based artificial agent. In *Eleventh Natl. Conf. on AI*, pp. 824-829.

[7] Krotkov, E. and Hebert, M. 1995. Mapping and positioning for a prototype lunar rover. *Proceedings of IEEE Int'l Conf. on Robotics and Automation*, pp. 2913-2919.

[8] Matthies, L., Gat, E., Harrison, R., Wilcox, B., Volpe, R., Litwin, T. 1995. Mars microrover navigation: performance evaluation and enhancement. In *Proceedings of IEEE Int'l Conf. on Intelligent Robots and Systems*, 1:433-440.

[9] Pagnot, R. and Grandjean, P. 1995. Fast cross-country navigation on fair terrains. In *Proceedings of IEEE Int'l Conf. on Robotics and Automation*, pp. 2593-2598.

[10] Rosenblatt, J. and Thorpe, C. 1995. Combining multiple goals in a behavior-based architecture. In *Proceedings of IEEE Int'l Conf. on Intelligent Robots and Systems*, 1:136-141.

[11] Santos-Victor, J. and Sandini, G. 1995. Uncalibrated obstacle detection using normal flow.

[12] Santos-Victor, J., Sandini, G., Curotto, F. and Garibaldi, S. 1995. Divergent stereo in autonomous navigation: from bees to robots. *Int'l Journal of Computer Vision*, pp. 159-177.

[13] Simmons, R., Krotkov, E., Chrisman, L., Cozman, F., Goodwin, R., Hebert, M., Katragadda, L., Koenig, S., Krishnaswamy, G., Shinoda, Y., Whittaker, W., and Klarer, P. 1995. Experience with rover navigation for lunar-like terrains. In *Proceedings of IEEE Int'l Conf. on Intelligent Robots and Systems*, 1:441-446.

[14] Turk, M.A., Morgenthaler, D.G., Gremban, K.D., and Marra, M. 1988. VITS - a vision system for autonomous land vehicle navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):342-361.

[15] Ullman, S. 1984. Visual routines. *Cognition*, 18:97-159.

[16] Zeng, N., Crisman, J.D. 1995. Categorical color projection for robot road following. In *Proceedings of IEEE Int'l Conf. on Robotics and Automation*, pp. 1080-1085.