Visually-Guided Obstacle Avoidance in Unstructured Environments

by

Liana M. Lorigo

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1996

© Massachusetts Institute of Technology 1996

Signature of Author .....
Department of Electrical Engineering and Computer Science
January 31, 1996

Certified by .....
Rodney A. Brooks
Professor
Thesis Supervisor

Certified by .....
W. Eric L. Grimson
Professor
Thesis Supervisor

Accepted by .....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
APR 11 1996
Eng.
LIBRARIES

Visually-Guided Obstacle Avoidance in Unstructured Environments

by

Liana M. Lorigo

Submitted to the Department of Electrical Engineering and Computer Science
on January 31, 1996, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Potential applications of autonomous mobile robots span a wide range. Some applications involve navigating a known environment, such as a robot delivering coffee in an office for which it has a complete map. Other applications preclude the existence of a map, such as a robotic exploration of the surface of the planet Mars.

This thesis presents an autonomous vision-based obstacle avoidance system which neither assumes nor stores knowledge of the environment or obstacle locations. The system consists of several independent vision modules for obstacle detection, each of which is computationally simple and uses a different criterion to localize the obstacles in the image. The system then automatically selects the modules needed to direct the robot. Selection proceeds exclusively from the outputs of the individual modules; no preferences are pre-determined based on the environment or pre-specified by the user. The vision modules in the current system are based on brightness gradient magnitudes, RGB (Red, Green, Blue) color information, and HSV (Hue, Saturation, Value) color information. The combination of modules is robust to a much wider variety of environments than a single module. The system is implemented on a small mobile robot with a single camera and uses very low resolution images. It has been tested for over 100 hours in diverse environments.

Thesis Supervisor: Rodney A. Brooks
Title: Professor

Thesis Supervisor: W. Eric L. Grimson
Title: Professor

Acknowledgments

This document describes research performed at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. Support was provided by Jet Propulsion Laboratories contract 959333 and a National Science Foundation graduate fellowship.

I thank my thesis advisors Rod Brooks and Eric Grimson for providing invaluable insight, ideas, and support, while giving me the freedom to shape my research. I have learned much from their expertise and am grateful for the time they have devoted to me. This work has benefited from many discussions with Ian Horswill about computer vision and robots. I thank him for his enthusiasm, patience, and advice. I specially thank my undergraduate advisor Dan Huttenlocher for his terrific guidance over the past several years.

I thank the people at IS Robotics and James McLurkin for their help with Pebbles throughout this project, Chris Barnhart for his help with the CVM and for providing CVM libraries, and Anthony Johnson for his help with specific experiments.

I also thank my many colleagues and friends here at the AI Lab and elsewhere who have contributed to this research through discussion and friendship.

Finally and most importantly, I thank my family – Charles, Shirley, Susan, and Lori Lorigo – for everything.

Contents

Chapter 1

Introduction

Potential applications of autonomous mobile robots span a wide range. Some applications involve navigating a known environment, such as a robot delivering coffee in an office for which it has a complete map. Other applications preclude the existence of a map, such as a robotic exploration of the surface of the planet Mars.

This work addresses the problem of designing a mobile robot to avoid obstacles while traveling in unstructured environments, that is, environments for which no knowledge of the appearance of the ground or the locations or appearance of the obstacles is available.

1.1 A Modular Avoidance System

An autonomous obstacle avoidance system comprised of multiple independent vision modules for obstacle detection is presented. Obstacle detection operates on single images only, and no map or memory of previous images is created or stored. Each module is computationally simple and uses a different criterion to localize the obstacles in the image. The system automatically selects the modules needed to direct the robot. Selection proceeds exclusively from the outputs of the individual modules: no preferences are pre-determined based on the environment or pre-specified by the user. The vision modules in the current system are based on brightness gradient magnitudes, RGB (Red, Green, Blue) color information, and HSV (Hue, Saturation,

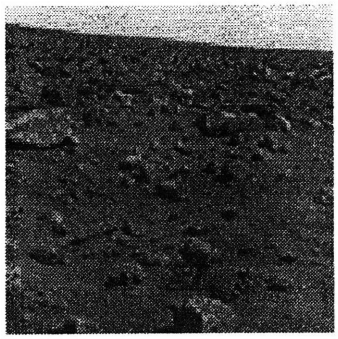

Figure 1-1: Image of Mars surface from Viking 2 lander.

Value) color information. The system is implemented on a small mobile robot with a single camera and uses very low resolution images of 64 x 64 pixels.

The system does not perform high-level navigation, such as moving toward a predefined target. Instead, it performs obstacle detection and avoidance, so that an autonomous robot can travel safely in cluttered, unstructured terrain, transmitting images, collecting samples, or performing some other task. Other researchers may make use of this work in designing their own robots, or in simply using these algorithms, so that they may investigate other areas of robotics and vision without worrying about obstacle avoidance.

1.2 Environments

One motivating goal of this research is the development of a robotic system to explore the Martian surface (shown in Figure 1-1) by using visual cues to avoid rocks and craters too large to traverse. Some of the settings in which the system was tested – rough outdoor areas and an indoor room of rocks and gravel – were motivated by this surface. The system was also tested in several other indoor settings, such as lounges, corridors, and offices. The background, lighting, and amount, type, and distribution of clutter varied between runs. These settings contain different types and textures of floor and carpeting.

1.3 Hardware Issues

For interplanetary exploration, it is important that robots be small, consume little power, and perform all processing on-board. Size is critical because larger and heavier equipment is more expensive to send into orbit than smaller, lighter equipment. Accordingly, the robot used in the current obstacle avoidance system weighs only 16 pounds and could weigh less as much of the equipment on it is unused. Low power-consumption is important because the weight of the batteries is a factor in the price of interplanetary missions. All vision processing in the current system is performed on vision hardware that consumes a maximum of 15 watts. Finally, it is desirable that all processing proceed on-board the robot because the light delay from the Earth to Mars, for example, is tens of minutes; thus, the possibility of remote operation is limited. Accordingly, all processing proceeds on-board the robot in the current system.

1.4 Summary of Results

Successful obstacle avoidance behavior is observed in diverse cluttered environments using the current system implemented on a small mobile robot. The system is comprised of multiple vision modules, and the combination of modules is robust to a much wider variety of environments than a single module. Test environments for the system include a room in which the floor is rough gravel and the obstacles are larger rocks of various sizes, and a carpeted room in which the obstacles are chairs, bookshelves, walls, and people. Preliminary outdoor experiments have also been performed. The system has been run cumulatively for over 100 hours. Pictures of the robot in operation are shown in Figure 1-2. Additional pictures are shown in Chapter 4.

1.5 Roadmap

Chapter 2 describes the work from computer vision, robotics, and artificial intelligence upon which this research builds. Chapter 3 presents the proposed and implemented

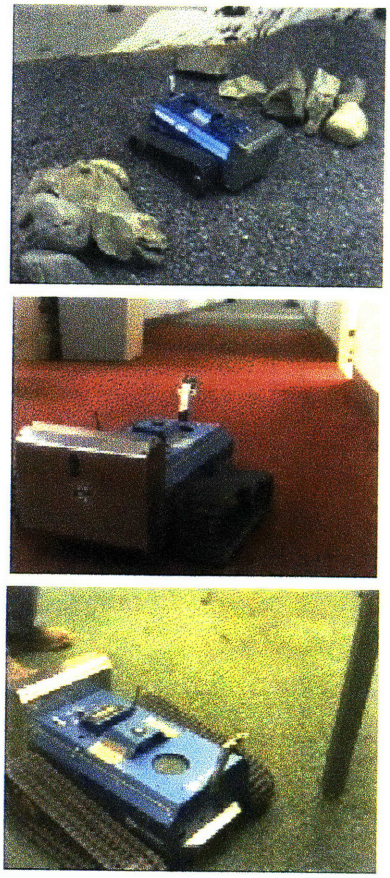

Figure 1-2: Autonomous obstacle avoidance system in operation on a mobile robot.

visually-guided obstacle avoidance system. It is comprised of multiple independent visual processing modules, each of which corresponds to obstacle detection in an environment that exhibits particular properties. The modules are explained in terms of a common framework shared across the modules and the actual computations that differ among them. The approach for combining the outputs of the individual modules and computing the motor commands is then detailed. Chapter 4 describes results of testing the system in various environments and categorizes failure modes. The discussion of the system concludes with remarks on its robustness to environmental and implementation variations. Related approaches that were initially explored but not incorporated into the current system are then discussed. Finally, Chapter 5 suggest areas of future work, and Chapter 6 draws conclusions about the research discussed herein.

Chapter 2

Related Work

This thesis draws on earlier work in the fields of robotics, computer vision, and control strategies. Presented below is a brief discussion of some of the work from these fields that is directly related to this work.

2.1 Robotics and Cooperation

In the design of any autonomous system that interacts with the world, a decision must be made about how much of the world to represent in memory and how much to compute directly from the sensory input at each time step. Strategies vary along the spectrum from storing large-scale world models to using the sensors exclusively. The latter approach, in which no world model is stored, is called *reactive*. The current work follows this strategy in that little memory of past obstacles or images is stored.

An important feature of the current system is that the visual processing is performed by multiple independent modules whose outputs are combined to yield the overall functionality of the system. There have been several general strategies for the cooperation of disparate modules within a single system.

The subsumption architecture advocates the decomposition of a system in terms of its task-achieving behaviors rather than in terms of the functional units of the system (Brooks 1986). Examples of task-achieving behaviors for a mobile robot are obstacle avoidance, exploration, goal-directed navigation, and reasoning, while examples of

functional units are perception, modeling, planning, and motor control. Systems using the subsumption architecture are comprised of layers of increasingly complex behaviors and have no central planning unit. Each behavior is self-contained and extends from perception to action. Following this approach, the current design has no central planner and consists of two layers of behaviors.

Since the advent of the subsumption architecture, several related approaches have been investigated. One example advocates minimizing the information loss that occurs between layers of a subsumption architecture-based system (Payton, Rosenblatt & Kersey 1990) (Rosenblatt & Payton 1989). Toward this end, "fine-grained behaviors" are described which minimize the amount of processing obscured by each module, making intermediate processing steps available outside of the given module. The current system uses this idea in combining the outputs of the various vision modules. This body of research also describes "internalized plans", which lie between reactive control and full planning. Plans are represented just as any other module so that the hierarchical ordering of modules is less important than in a full-planning system, and so that other modules may interact with the plan to choose an optimal strategy. To incorporate high-level navigation into the current obstacle avoidance system, it would be straightforward to represent the higher-level goal of moving toward a target, for example, as such an "internalized plan."

Rosenblatt & Thorpe (1995) describes DAMN, an architecture for integrating information from various routines. This architecture is flexible in that it takes a set of weights as input which determines the relative importance of each module in the decision-making process. These weights can be set by the user depending on the system's objective and the current terrain.

2.2 Specialization

Prior knowledge about the domain in which a system is to operate is useful in its development. Algorithmic simplifications can be made according to such constraints. When such simplifications are made explicit, the system's behavior in a new do-

Figure 2-1: When the ground is flat and all objects rest on the ground, the ground plane constraint implies that nearby objects appear lower in the image than distant obstacles. Notice that the close rock on the right side of the image appears lower than the distant rocks.

main can be predicted. The term *specialization* refers to the formal theory that supports these ideas. Specialization, as described by Horswill (Horswill 1993*b*) (Horswill 1993*a*), is used in the development of the current system.

Given some hypothesized general agent, one can derive an actual specialized agent that exhibits a target behavior(s) in a target environment(s). The derivation consists of a series of transformations. Each transformation is paired with an environmental property. Transformations are simplifications which carve away the space of all possible tasks to the space of the particular task the agent is intended to accomplish.

The series of constraint-transformation pairs provides two definitions. The constraint components define the domains in which the agent can operate; that is, its success is predictable in untested environments satisfying the constraints. The transformation components define an algorithm that will achieve the desired behavior. To extend, change, or reproduce a specialized system, one omits the transformations that correspond to properties no longer present and adds transformations that correspond to new properties.

2.2.1 Example

The following constraint-transformation pair was used by Horswill and is also used in this work.

Consider the task of avoiding obstacles during autonomous visually-guided navigation. At any given time, a goal is to determine the nearest obstacle so the robot

may avoid it. Assuming that the robot is operating on a flat surface and that all obstacles rest on the ground, the *ground plane constraint* implies that image height of obstacles is in a monotonic relation to scene depth. That is, nearby objects appear lower and distant objects appear higher, as illustrated by Figure 2-1. The ensuing simplification is the use of image height instead of a more complicated measure of scene depth to determine the nearest obstacle. This obstacle is found by scanning the image from the bottom to the top until an object is encountered.

2.2.2 Agent-Environment Interactions

Specialization is one technique for building systems that rely on underlying interactions between the systems and their environments. Many researchers have studied agent-environment interactions more generally. In such work, the term "agent" can be used to refer to any independent unit of processing, from a program component to a mobile robot. One major idea is that the distinction between the agent and its environment is arbitrary. For example, a motor of a robot can readily be considered part of the environment of the robot's overall controller. Current research focuses on understanding agents, environments, and interactions as complete non-linear dynamical systems (Beer 1988). That is, three criteria must be met: 1) the agents, environments, and interactions have identifiable states, 2) state changes are continuous, and 3) such changes are captured by a set of (non-linear) functions that define a vector field over the state space (Smithers 1995).

The work discussed above is independent of the sensor used by a system. For the current work, the environment is sensed by vision exclusively, and the vision algorithms build on the computer vision research discussed in the next section.

2.3 Computer Vision

Computer vision is an active research area because vision is a data-rich sensor and therefore is useful for a wide variety of tasks. This section discusses sub-fields of computer vision that motivated the current work, including visual routines theory,

animate vision, visually-guided robotics, optical flow, road and off-road navigation, and research specifically designed for use on Mars or the Moon.

2.3.1 Visual Routines

Visual routines theory suggests that visual systems have three components. Some small fixed number of registers, or *markers*, store image locations corresponding to various types of visual information. Some *mechanism* directs visual attention to the portions of the image that are relevant to the current task. *Primitive image operations*, such as coloring and edge-detecting, can be combined as in a computer program to process the image (Ullman 1984) (Horswill 1995). A machine vision system comprised of these elements is referred to as a visual routine processor (VRP). This theory has received attention from both the planning community and the vision community. Agre and Chapman's implementation of a VRP used to play a video game was the first well-known implementation, although they bypassed vision and interfaced the "sensor" directly to a world model (Agre & Chapman 1987). Mahoney has defined and implemented a visual programming language in which elementary operations are combined to form visual routines (Mahoney 1993) (Mahoney forthcoming). His language uses the operations suggested by Ullman as well as additional operations and has been used for document processing. The first implementation of visual routines theory on real-time video is discussed in (Horswill 1995). The current system is related to the VRP model in that it performs a series of primitive image-based operations and only pays attention to task-relevant image regions, namely the boundaries between ground and the nearest obstacles. However, the current system does not combine the elementary operations according to the VRP model.

2.3.2 Animate Vision

A related strategy in the construction of computer vision systems is *animate* vision (Krotkov 1989) (Ballard 1991) (Coombs 1992) (Aloimonos 1993). Animate vision is the strategy of controlling the camera view based on previous visual sensing so

that subsequent computations may be simplified. Such systems therefore normally include a gaze control mechanism (Pahlavan & Eklundh 1992). Initial processing is performed quickly at coarse resolution to determine an appropriate camera position. The camera is then repositioned and further image computations are performed. For example, if a camera is positioned so that a particular target remains at the center of its field of view, then the task of segmenting the target from the image is easier since its position is known.

2.3.3 Most Similar Visually-Guided Robotics

The current system is most closely related to Horswill's research with the vision-based robot Polly (Horswill 1993*b*) (Horswill 1993*a*). Polly's task was to roam the corridors and rooms of MIT's AI Laboratory giving tours to visitors. The entire system performed a variety of behaviors. from low-level obstacle avoidance and corridor-following to interacting with people. It relied on relatively simple algorithms, an approach termed "lightweight vision". The current system maintains these ideals but addresses a more general obstacle avoidance problem, although it does not perform higher-level navigation.

Polly's algorithms used known properties of its environment to simplify the problem of vision-based obstacle avoidance and navigation. Since its environment was assumed to have a textureless floor. the problem of detecting obstacles became simply the problem of finding edges. Further, the ground plane constraint implied that an image height computation could be substituted for a more difficult depth computation. The presence of well-defined corridors simplified the problem of determining the robot's direction. These principles are currently being incorporated into many commercial applications, including a semi-automatic visually guided wheelchair, an autonomous office messenger robot. and other systems (Gomi 1995*a*) (Gomi 1995*b*) (Gomi, Ide & Maheral 1995) (Gomi, Ide & Matsuo 1994).

Gavin worked on extending the approaches of lightweight vision to handle the case in which the environment is the surface of Mars (Gavin 1994). He analyzed processing costs and hardware requirements for a set of separate vision routines. He concluded

that this approach is indeed appropriate for an autonomous visually-guided robot designed to explore the Martian surface.

2.3.4 Optical Flow-Based Depth Reconstruction

Several related robotic systems use optical flow to obtain depth information about their environments. This section introduces the concept of optical flow then presents some systems that use it.

Optical flow is the use of temporal variations in image intensity patterns to extract information about relative motion. The optical flow field is a two-dimensional field of vectors (u, v) corresponding to the apparent motion of these patterns over a sequence of frames (Horn 1986). If total image brightness remains constant over time, the field is given by:

$$E_x u + E_y v + E_t = 0$$

where E_x, E_y, and E_t are the horizontal, vertical, and temporal derivatives of image brightness, respectively, and the motion component is given by the vectors (u, v). This equation states that the perceived motion of brightness patterns within the image is due to actual motion of the scene and not due to varying illumination conditions. This flow field gives the motion of objects in the image with respect to each other or with respect to the camera. In the case of a moving camera in a motionless scene, depth automatically follows from image motion because motion parallax implies that nearby objects will appear to move more than distant objects. Hence, its importance for robotic obstacle avoidance is that if the motion of the robot's camera is known and the environment is static, optical flow gives the depths of potential obstacles. Limitations of optical flow include the requirement of rich visual texture, the precision to which motion must be known, and the large number of iterations normally required to compute the flow accurately. While the first two limitations are inherent to optical flow, the computational difficulty can be reduced if the motion can be constrained (Horn & Weldon 1988).

Iterative approaches to acquiring depth from motion focus on general depth re-

construction. One such approach uses Kalman filters (Matthies, Szeliski & Kanade 1988). The system described by this work differs in that it maintains little or no information from previous frames whereas the Kalman filter system stores previously computed depth maps and associated uncertainty values. The current system does not address a general depth from motion problem, but rather uses only a simple approximation to depth designed to address the task of obstacle avoidance. However, for particular platforms and applications, this temporal approach to depth perception is appropriate. Combined with spatial information, a temporal approach has been used in a successful road-following system which attains very high speeds and robustness to some environmental changes (Dickmanns, Mysliwetz & Christians 1990).

Related work in visually-guided robotics has used a simplification of optical flow for obstacle avoidance (Coombs & Roberts 1992) (Santos-Victor, Sandini, Curotto & Garibaldi 1995). Only the flow in the direction of the normal to the image intensity gradient is computed. This flow is called *normal flow* or gradient-parallel optical flow, and it is the only flow component that can be computed locally. Biological evidence motivates the use of normal flow for navigation. Bees navigate using specialized visual-sensing cells that can only detect brightness gradients along a particular orientation (Srinivasan, Lehrer, Kirchner & Zhang 1991) (Horridge 1986) (Franceschini, Pichon & Blanes 1991). Further, because the eyes of bees are aimed sideways, a bee is equidistant between obstacles to its right and to its left exactly when the normal flow is equal between its two eyes. Following this work, Santos-Victor et al. (1995) has built Robee and Coombs & Roberts (1992) has built Beebot. Both robots use the technique of equalizing normal flow between two divergent cameras. A related monocular robotic system compares the expected flow of a flat ground plane to the perceived flow in the images (Santos-Victor & Sandini 1995). Inconsistencies are considered to be obstacles as they lie outside the ground plane.

2.3.5 Road and Off-Road Navigation

A related project in outdoor navigation is the UGV (Unmanned Ground Vehicle) (Hebert, Pomerleau, Stentz & Thorpe 1995). It maintains a "traversability map"

according to range data for obstacle avoidance, combines vision and a large neural network for road following, and chooses behaviors by a Gaussian distribution-based arbiter (Rosenblatt & Thorpe 1995).

The Autonomous Land Vehicle (ALV) project is a large-scale project that combines advanced hardware architectures, computer vision, autonomous navigation, and other areas of artificial intelligence (Turk, Morgenthaler, Gremban & Marra 1988). VITS, the ALV's vision subsystem, uses both a color video camera and a laser range scanner for obstacle avoidance and road-following. The goal of the video system is to segment the image into "road" and "non-road" regions. Animate vision is used to ensure that the road is approximately in the center of the image. Various segmentation algorithms are then used, including a "Red minus Blue" algorithm. Consider a road scene in which the road is paved and the off-road region is dirt. In an RGB representation of such a scene, the pavement is observed to have a strong blue component, but a weak red component. The dirt, however, is observed to have a strong red component, but a weak blue component. By subtracting the blue band from the red band, the intensity difference between the pavement and the dirt is emphasized, simplifying segmentation. After reducing the original image to a binary image of road and non-road pixels, a road boundary is extracted and transformed into three dimensional coordinates which are incorporated into a vehicle-centered world model. Motor commands are generated from this model.

Another road-following project which relies on vision uses the SCARF classification algorithm (Crisman 1991). The SCARF algorithm clusters image pixels according to some image property. This system is effective, but, like the UGV, can require intensive computation. Particular features, however, can be chosen for efficiency. For example, a more recent system uses one-dimensional "categorical" color instead of three-dimensional RGB color (Zeng & Crisman 1995).

A behavior-based off-road navigation system is described by Langer, Rosenblatt & Hebert (1994). This system uses range-data for depth perception and operates on a military vehicle. The task is not higher-level navigation but is to avoid untraversable regions while traveling in rough, unmapped terrain. Its success has been demonstrated

in an experiment in which it traveled autonomously for one kilometer in such terrain. Untraversable regions are determined by converting the output of the range image to an elevation map. This information is passed to a map management module that maintains a local map of the terrain which is used by a planning module to issue the motor commands.

2.3.6 Research for Mars and Lunar Environments

Several groups of researchers have investigated autonomous navigation with Mars or the Moon as the target environment. Some groups have used vision while others have focused on other sensors. Many of the vision-based systems for this task rely on stereo vision.

Like this work, Pagnot & Grandjean (1995)'s work in cross-country navigation for Mars or the Moon emphasizes a modular design. Their approach is somewhat reactive in that it distinguishes between two modes of operation: a "full planning" mode, and a simple mode which makes some assumptions about the terrain. They require off-board processing for depth perception by either a robust stereo method or a laser range finder. Then, using known depth values, they address the problem of navigation. The current work does not address the issue of higher-level navigation but for obstacle avoidance uses a combination of simpler but less reliable on-board algorithms to determine depth.

Experiments with the Ratler robot, a prototype lunar rover, also use stereo vision algorithms for depth perception (Krotkov & Hebert 1995) (Simmons, Krotkov, Chrisman, Cozman, Goodwin, Hebert, Katragadda, Koenig, Krishnaswamy, Shinoda, Whittaker & Klarer 1995). The stereo component of the system produces terrain elevation maps which are used by the obstacle avoidance component for path planning. These experiments have been very effective at obstacle avoidance but require much compute power and off-board image processing.

Work at Jet Propulsion Laboratories (Matthies, Gat, Harrison, Wilcox, Volpe & Litwin 1995) on Mars rover navigation focuses on reliable user-assisted navigation for upcoming Mars exploration missions. Their system does not use passive vision;

rather it uses active triangulation from a light-striper, in conjunction with several other sensors for reliability. This current research in passive vision can potentially be used to reduce the energy consumption on future Mars rovers.

At Draper Laboratories, Lynn explored the choice of a one-camera system using correlation-based depth from motion for a Mars Micro-Rover robot (Lynn 1994). Correlation was implemented in hardware for efficiency, and he compared this approach with gradient-based optical flow. He concludes that the pattern matching approach offers equally good results as a gradient-based flow method at a fraction of the computational cost. However, the system had been tested only in a structured indoor setting.

2.4 Summary

The approach described by this work for designing an autonomous obstacle avoidance system builds upon the work described above in vision, robotics, and artificial intelligence. The vision algorithms emphasize specialization and computational simplicity and are intended to operate in a wide variety of unstructured environments. Further, the approach maintains the ideals of reactive systems and incorporates previously developed techniques for combining multiple processing modules into a single system.

Chapter 3

A Modular Avoidance System

This work presents an autonomous obstacle avoidance system that incorporates computer vision, robotics, and control strategies. In this chapter, its design and implementation on a mobile robot are described.

The high-level architecture of the system is illustrated in Figure 3-1. Each visual processing module takes as input the image frame from the robot's camera and generates a depth map. These maps are combined into a single depth map which is converted to motor commands. A motor control routine then checks if these commands indicate that the robot is "stuck" and sends the final motor commands to the motors.

The system's description presented in this chapter also follows Figure 3-1. After a description of the hardware platform, the components of the system are presented in the order defined by this structure. The chapter then analyzes the running time and concludes with remarks on the overall design.

3.1 Platform

The system is housed in the Pebbles III Robot, pictured in Figure 3-2, designed and built by IS Robotics. Inc. (IS Robotics 1995)[1]. Pebbles' sturdy design, especially the

[1]Some hardware modifications have been made to Pebbles for this project, including repositioning the camera.

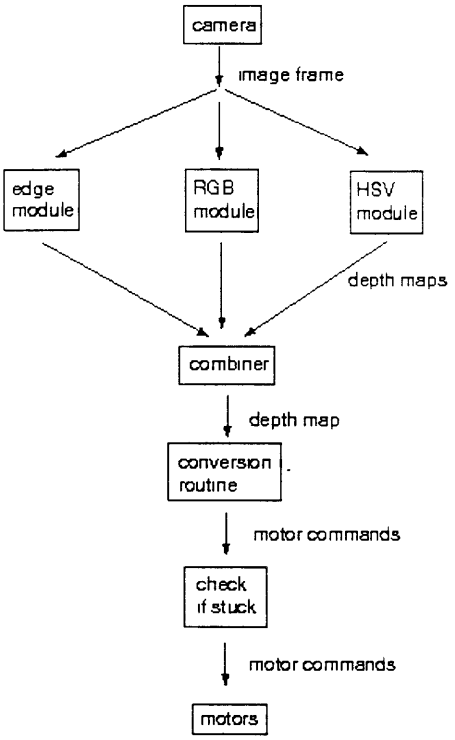

Figure 3-1: Control flow of current obstacle avoidance system.

tracked rather than wheeled base, makes it well-suited for travel in rough terrain. Pebbles is equipped with a Motorola 68332 processor, a Chinon 3mm camera positioned at the front of the robot 10.5 inches off the ground, and a video transmitter. For this work, the transmitter is used only to view processed images, and all processing is performed on-board the robot. Furthermore, the system uses a visual-processing hardware system, the CVM (Cheap Vision Machine), designed by DIDEAS, Inc. The processor is a Texas Instruments C30 DSP. The CVM holds 1MB of memory and supports high-speed video input. The vision software is written in C and runs on the CVM. The system runs a smaller control program, written in L (a subset of LISP) (Brooks 1994) on the 68332.

3.2 Vision Modules

The system is comprised of multiple independent vision modules. The purpose of each module is to generate a relative depth map of obstacle locations based on the input

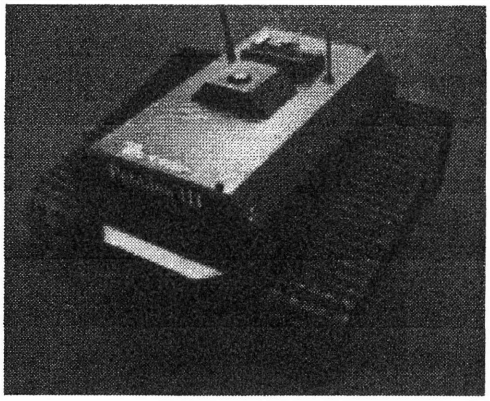

Figure 3-2: The Pebbles III Robot.

image frame. For each module, this map is stored in a one-dimensional array, called a "depth array," which is indexed according to the horizontal axis of the image. For each x-coordinate of the image, this array stores the relative distance from the robot to the nearest obstacle at that horizontal location, as determined by the particular module. Eventually, the depth arrays from the modules are combined into a single array which is used to determine the motor commands. Depth information is used to determine the direction in which the robot should turn, based on which side (left or right) of the image has more free space ahead. The rest of this section discusses the modules' common framework and the particular properties detected by each module.

3.2.1 Framework

The generalized module takes as its input an image, scans a window over the image, and for each location of the window computes the value of the module-specific property. Possible properties measured include average color and histograms of intensities. If the measure of the window according to a module is different from that module's notion of "safe ground," then it indicates that an obstacle has been detected. Note that different modules may find different obstacles since they use different measures for detection purposes.

The environments in which the system is intended to operate exhibit several regularities: the ground type at the robot's initial location will determine the type of ground it considers favorable, boundaries between safe ground and obstacles are visi-

ble in single image frames. the ground is nearly flat, and all objects rest on the ground. These regularities, or domain constraints, directly lead to the structure shared by each module. In each module a different measure is computed on the image. This metric is the distinguishing variable of each module. This section presents this structure, with each step motivated by its corresponding domain constraints.

The first domain constraint involves the position of the camera on the robot and the ground on which the robot begins its movement. The system assumes that the camera is positioned at the front of the robot and is aimed forward. although it can be tilted. Also, it assumes that the robot starts out on favorable, safe ground and that this safe ground extends far enough ahead of the robot to be visible by the camera. These assumptions imply that the region seen in the lowest portion of the image is a favorable type of surface. as it corresponds to the safe ground immediately in front of the robot. So if the robot stays on that same surface type, it will remain safe. Accordingly, each module computes and stores some "measure" of the bottom portion of the image. This will be the measure it strives to maintain, due to the assumption that all image regions with this measure correspond to the same type of ground as the bottom portion of the image which represents safe ground. This measure of "safe" is recomputed for each image frame and therefore can be considered an initialization step for each frame: the vision modules store no memory across frames.

The purpose of the modules is to determine the relative distance from the camera to the nearest obstacle for each x-coordinate of the image. The domain constraints that the ground is nearly flat and that all objects rest on the ground imply that, for the most part, closer obstacles appear lower in the image than farther obstacles; thus the image height of an obstacle can be used as an approximation to relative depth. Together with the constraint that obstacle boundaries are visually detectable in single image frames. these constraints imply that the system can detect the nearest obstacle in an image column (x-coordinate) by starting at the bottom of the column and scanning upward until a visual boundary is detected. It then proceeds to the next x-coordinate. Note that this method only finds the obstacle boundaries closest to the robot and does not find the distant borders of obstacles.

Figure 3-3: Image from Pebbles' camera, illustrating a vertical slice and a window.

The windows used to scan the image cannot span the entire width of the image, as this would not give information about the horizontal locations of obstacles which is necessary to compute the turn-angle of the robot. Instead, windows (and vertical image slices) are 20 pixels wide. Recall that the total image resolution is 64 x 64. For each slice, the module-specific measure is computed over the bottom window of that slice, where a window is 20 pixels wide and 10 pixels high (Figure 3-3). The module then scans up the slice one pixel at a time and recomputes the measure on each higher window. When the measure changes significantly from the stored measure of the bottom window, the module decides that it has encountered an obstacle boundary. It stores the height at which this change occurred at the index corresponding to the x-coordinate of the vertical slice, in the depth array. Specifically, the height of the bottom of the window is stored at the index corresponding to the x-coordinate that is the middle of the window.

The modules look at larger windows of the image instead of just single pixels to account for visual cues that are larger than a single pixel. For example, if the safe area (the floor) consists of a striped, two-tone carpet, a single pixel can only record one tone and therefore gives an incomplete description of the safe region. Looking at an area of pixels rather than a single pixel allows the modules to handle more complex environments.

Each module only shifts the slices horizontally by one pixel at a time to obtain the next slice so it acquires a depth (image height) value for each horizontal coordinate. Moreover, when scanning up the individual slices a module only shifts each window vertically by one pixel to obtain the next window so the depth values are found at

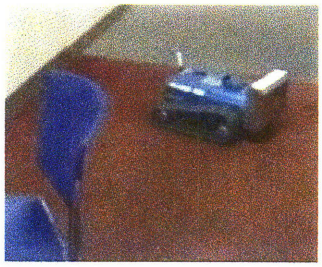

Figure 3-4: The carpet boundary is perceived as an obstacle since only single image frames are used for obstacle detection.

single-pixel precision.

This idea of scanning up image slices is motivated by Horswill's work with the robot Polly (Horswill 1993*a*). The difference is that, with Polly, "windows" were single pixels, and the "measure" was whether or not an edge of at least a particular image-dependent threshold was present. This simpler computation was appropriate because Polly assumed that the safe ground would not have sharp edge gradients and that the visual cue representing safe ground was accurately captured by a single pixel. These domain constraints implied that an edge gradient in the image indicated an obstacle. Since the current system relaxes these constraints, a more general approach is needed. Hence, the system computes over larger image portions and uses multiple measures to detect obstacles.

One drawback of the approach of scanning up single image frames to detect obstacles is that changes in the color or texture of the ground are perceived as obstacles. For example, boundaries between two different colors of carpet are perceived as obstacle boundaries, as pictured in Figure 3-4. In this case, the robot considers the different carpet an obstacle and avoids it. Additional sensors, multiple sequential images, or explicit knowledge of such boundaries would be required for a monocular system to address this problem.

In conclusion, each module produces a one-dimensional array of image heights (depths) corresponding to the x-dimension of the image. Overlaying this "depth array" on the image shows where obstacle boundaries were detected across the image. Figure 3-5 shows a depth array from one module overlaid on the image from which it was generated.

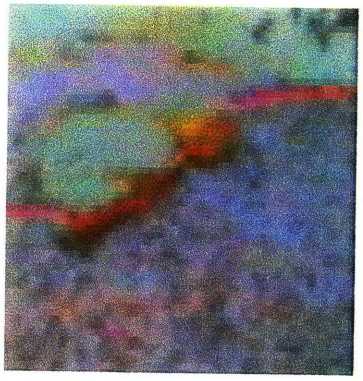

Figure 3-5: A depth map is shown in red overlaid on the image from which is was generated. This map was generated by the brightness gradient magnitude module of the system.

3.2.2 Module-Specific Measures

While the above constraints and subsequent algorithmic simplifications apply in all environments considered for this research, specific environments may satisfy additional constraints or impose restrictions on the solution. For example, obstacles may differ from the ground in various image properties, including intensity, color, brightness gradient magnitudes and orientations, and spatial frequency. These additional constraints are used to determine the measures which vary across modules.

The framework described in section 3.2.1 can be implemented with a variety of measures to yield many different modules. Any function that takes as input a window of the image and reproducibly outputs a result that can be numerically compared to a previous result is a candidate for such a measure. The comparison is used to determine if measures of higher windows are different than measures of the bottom window indicating an obstacle. Recall that the choice of a window of many pixels instead of a single pixel over which to compare allows the system to handle a wide variety of environments. In addition to this gain of functionality, this choice also achieves a gain in performance since single pixels are notoriously noisy compared to large regions when evaluated with an averaging or distribution-based technique.

The measures in the current system are each the histogram of a different image property over a given window of the image. They take as input an image window, compute the value of the module-specific property at each pixel in that window,

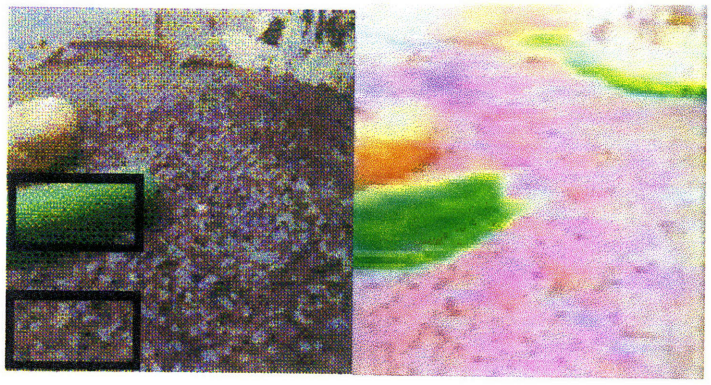

Figure 3-6: The sample histograms were generated from the first image. The specific windows corresponding to the histograms are outlined. The second image shows the complete depth array computed by the normalized RGB module for this image.

generate a histogram of the values, and output the histogram. Examples of values computed at individual pixel locations are color and brightness gradient magnitude. These values depend only on the current pixel or perhaps on some small number of neighboring pixels, as in brightness gradient detection. The histogram gives the distribution of values over the window without regard to the spatial arrangement of those values within the window. For histogram comparison, the metric used is the area between the histogram of the "safe" (bottom) window and the histogram of the current window. When this area is large, the current window is assumed to contain an obstacle.

For some measures, this single histogram approach is replaced by a pair of histograms when the value computed at a pixel is actually a pair of values, such as in normalized RGB and HSV color. The difference between windows is the sum of the differences in the areas under the corresponding histograms. For example, histograms corresponding to red and green colors at the obstacle boundary found in the leftmost vertical slice of the image in Figure 3-6 are shown in Figure 3-7.

Although many different modules have been considered, the current system is comprised of three modules. All three follow the framework above and use a histogram measure of various image properties. They differ only in what image property they compute at the individual pixels. That is, the histograms generated by different modules show the distributions of different image properties. The properties used in

the three modules are brightness gradient magnitude, normalized RGB (Red, Green, Blue) color, and normalized HSV (Hue, Saturation, Value) color, respectively. Other properties that work well are discussed in section 4.4.2. Note that each color value is a pair rather than a triple of numbers because the third component is redundant in normalized color. The system distinguishes between RGB and HSV color because they use different information to detect obstacles. Since the environments' illumination is not perceived as constant across even a region of an image, due to inter-reflections of objects and lighting positions, the system cannot assume constant intensity, even for a given region. For this reason, the observed color values are normalized.

Note that the entire histogram need not be recomputed 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. It must, however, save the initial histogram for each vertical image slice, corresponding to the bottommost window, as it is needed for comparison throughout the processing of that slice. The horizontal histogram shifts used to generate each histogram of the bottom window of a vertical slice from the histogram of the bottom window of the previous slice can be computed efficiently in the same way as vertical slices.

The brightness gradient magnitude module is outlined as an example. First, the grabbed image is blurred to reduce noise and converted from color to greyscale. Second, the magnitude of the brightness gradient at each pixel is computed as the average of the magnitude of its vertical brightness gradient and the magnitude of its horizontal brightness gradient. For efficiency, these magnitudes are approximated by the difference between the current pixel and the pixel above or alongside it, respectively. These magnitudes are normalized so that the maximum magnitude is 31. Note that, for standard 8-bit greyscale images, the maximum magnitude is 255. The 8-bit numbers are quantized to 5-bit numbers to accommodate sensing error. Third, the system follows the framework above with the histogram method, where the histograms are generated from the computed gradient magnitudes.

For the color modules, the approach is analogous, except that two histograms are

maintained at each comparison. In the case of normalized RGB color, red and green are used. For normalized HSV, hue and saturation are used. Recall that the third component is omitted in both color models since it is redundant.

3.3 Fusing the Modules

Each individual module outputs a depth array. After these arrays have been produced, the next step is to combine them into a single array from which the motor commands are generated. The system does not rely on any higher-level mechanism to supply the scheme for combining the depth arrays into the final one. Instead, the system uses the arrays themselves to determine their weighting in the final array. Two criteria used in the combination process are overall smoothness and pointwise median.

Obstacle boundaries in images usually appear smooth, with small variations between neighboring values in the depth array. This observation corresponds to the constraint that actual obstacle boundaries in the world are likely to be smooth. In the right half of Figure 3-8, notice that the smooth blue and green arrays much more closely approximate the actual obstacle boundary than the jagged red array.

This constraint of smoothness on obstacle boundaries suggests giving preference to modules producing smooth depth arrays. In this way, the first criterion is smoothness as a "confidence measure" of a depth array. Specifically, the system measures the smoothness of each array then assigns it a coefficient proportional to that smoothness. Smoothness is measured by taking second differences along the array. An overall depth array is produced by pointwise averaging the values in the individual depth arrays, weighted by the smoothness coefficients.

As an example, consider the images in Figure 3-8. The right half of the image shows the depth arrays of the individual modules overlaid on the image. Notice that the red array, which corresponds to the brightness gradient magnitude (edge) module, is jagged, while the blue and green arrays, corresponding to normalized RGB color and normalized HSV color respectively, are smoother. The red array on the left half of the image is the array computed by combining the original arrays according to this

smoothness-based approach. That is, the red array is the linear combination of the arrays from the individual modules, where each module is weighted according to its smoothness. The edge module earned a very low weight for this frame, since it was very discontinuous, and the other modules were more heavily weighted.

The second criterion for combining the depth arrays from the three modules relies on choosing the median value at each x-coordinate. When the majority of the modules are exhibiting good behavior, this median approach succeeds as well as the smoothness approach. In testing, it has usually produced results nearly identical to the smoothness approach, as illustrated by the left half of Figure 3-8, in which the blue array is obtained by the median function along the three individual arrays pictured in the right half of the figure. In this figure, note that both approaches successfully disregard the incorrect outlying information from the output of the edge module. Although both combining methods are demonstrated by the figure, in an actual system only one method is used, so only one final depth array is necessary. Both are shown here for comparison.

With either strategy the computation within the individual modules does not include the final determination of motor commands. Instead of viewing the modules as indivisible units that process all the way to the motor commands, the system allows the intermediate computation of the depth array to be available to the merging procedure. This strategy is an example of "fine-grained behaviors" (Rosenblatt & Payton 1989), since the depth arrays computed by each module are available to the merging procedure instead of only motor commands which could be generated from these individual depth arrays.

3.4 From Depth Arrays to Motor Commands

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

image.

The motor commands are determined as follows. The image is conceptually split in half. Let sum_{left} be the sum of the values in the left half of the depth array; likewise for sum_{right}. It may be more intuitive to consider these sums as averages, that is, as the average depth of an obstacle in a given half of the camera's field-of-view. Constants can be ignored because hardware-dependent calibration coefficients are needed for the actual motor commands, so these numbers are left as sums for efficiency.

First, the system computes the desired turn angle. If one side has significantly fewer obstacles than the other side, then the robot should turn sharply toward it. Else, it should not turn as sharply. Thus the turn angle is simply a difference:

$$turnangle = c_1 * (sum_{left} - sum_{right})$$

where c_1 is a constant dependent on the robot's motors. This equation may seem to imply that if the robot were moving straight toward a wall then it would go straight forward until it got "stuck" and turned in place. In practice, however, sensing error or a slight angle of motion makes this difference non-zero so the robot turns slightly. The difference for the next image frame is most likely to be increased due to this turn. This process continues until the robot has turned away from the wall as illustrated by Figure 3-9.

Next, the forward speed is computed. The robot should move quickly if obstacles are far away and should move slowly and cautiously if obstacles are too close, perhaps even move in reverse. The total distance to obstacles is computed by adding the sums of the two halves. Some constant k is subtracted to achieve reverse motion when obstacles are too close. Varying k changes how close the robot can get to obstacles before determining that they are "too close" and backing up. Hence, the equation is:

$$forwardspeed = c_2 * (sum_{left} + sum_{right} - k)$$

where c_2 is another motor calibration constant. c_1 and c_2 are required to calibrate

the approximations to the correct motor commands and are quickly determined by observing several arrays and desired motor commands and then solving for the coefficients. k is also determined in this way. Since Pebbles' field-of-view is insufficient to include objects very close to but off to the side of Pebbles, the coefficient of *turnangle* is exaggerated so that the turn does not result in a situation where the obstacle is invisible. Rather, the out-of-view obstacle is completely cleared by a sharper turn.

Finally, a simple conversion yields left and right motor commands, which are the only commands necessary for a tracked robot:

$$command_{left} = forwardspeed - turnangle$$

$$command_{right} = forwardspeed + turnangle$$

This method of reducing the image to a one-dimensional depth array, then averaging over regions to yield motor commands was suggested by Horswill's work with his robot Polly. He divided the array into three segments to yield values intuitively similar to sum_{left}, "sum_{center}," and sum_{right}. The turn angle was computed as the difference of the side values, but the speed was computed by using only the center value. This made sense because Polly's field-of-view was significantly wider than its width so that only obstacles detected in the center of its image would be hit if it proceeded forward. Pebbles' field-of-view to width-of-robot ratio, however, is such that even obstacles detected in the periphery of its images would be hit by its proceeding directly forward. It follows that the entire image is needed for the forward speed computation.

The loop of the current system concludes with a supervisory motor control routine that checks if the robot is "stuck," according to the motor commands suggested by the vision modules. That is, it determines if the commands indicate that there is no free space ahead of the robot by checking if they are below a threshold. If so, the robot is commanded to turn in place until the vision program issues motor commands indicating a clear path ahead, at which time the vision program again regains control and its motor commands are followed. Note that the vision modules

have no knowledge of whether or not the robot is stuck: they detect obstacles solely based on the current image. The supervisory routine operates independently of the vision modules.

3.5 Efficiency Analysis

The system's running time per image frame depends on the image resolution, the window size, and the range of the function whose values define the histogram(s). In the current implementation, the speed of the robot is approximately 0.3 meters/second, and the processing speed is approximately 4 frames/second.

Let w and h be the width and height in pixels of the image, respectively. Let m be the width of the window that is scanned over the image and n be the length of the histogram that is generated for each window. Assume that there is a fixed number of modules. The running time of the obstacle avoidance system is $O(wh(m+n))$, explained as follows.

- Initial processing is performed to convert the input image into an image containing the values from which the histograms are generated. Examples of such processing steps are blurring, detecting gradients, and conversions to normalized RGB and normalized HSV. Since such operations rely only on a single pixel or a small fixed-size neighborhood of pixels, each runs in $O(wh)$.

- To generate the histograms for each module, a window is scanned up each column of the image. For each column, a row of the window is added and subtracted at each shift. There are h shifts per column, w columns, and m pixels per row, so generating the histograms is $O(whm)$.

- For each column, the current histogram is compared to the initial histogram at each upward shift of the window. Each comparison is a pointwise difference of the histograms so takes $O(n)$. Thus the total time for comparing the histograms of an image frame is $O(whn)$ for each module.

- Combining the modules involves scanning the depth arrays a small fixed number of times and is thus $O(w)$.

- The routine that checks if the robot is stuck requires constant time.

3.6 Design Attributes

The most important feature of this system is that it runs multiple, simple, independent visual processing modules on each image. The outputs of the individual modules are combined to produce the motor commands. The weights assigned to the outputs to execute the combination depend on the current environment in that they are derived directly from the outputs themselves. Each visual module exploits only one image property to roughly segment obstacles from safe ground and therefore performs obstacle detection in a domain that exhibits that particular property better than the other modules.

No map or memory of past images is stored by the vision system. Rather, at each time step, new motor commands are issued based solely on the current image from the robot's camera. The system is *reactive* in that it only interacts with the environment as it is currently perceived by the system's sensors, as opposed to planning behavior according to a stored world model. This approach suffers when sensors are faulty or insufficient, so redundant sensors are advantageous. However, if the world is sampled frequently enough and in sufficient detail, then the system is not greatly affected by a small number of mistakes; since the errors are not stored, they pose no threat to subsequent processing, assuming uncorrelated errors.

The current system is behavior-based in that it uses relatively simple modular behaviors to exhibit increasingly complex behaviors (Brooks 1986). In short, the "stuck" layer subsumes the vision layer when appropriate. Future work could incorporate additional layers such as higher-level navigation capabilities. These strategies are examples of Brooks' subsumption architecture, in which behaviors are layered to achieve increased complexity.

Another distinctive feature of this system among visually-guided robots is the use

of only one camera for sensing the world. For unstructured environments, using two cameras for stereopsis is more common. Monocular systems are generally restricted to well-constrained environments and are often used in conjunction with non-visual sensors. The decision to use one camera is motivated by the desire for low-compute-power systems as mentioned in section 1.3. Note that this system cannot obtain the same type of information as a two-camera system (specifically, it cannot obtain binocular stereo), but it obtains enough information to perform obstacle avoidance in particular environments. In addition to the limitation of a single camera, the system assumes no camera calibration and uses very low resolution images of 64 x 64 pixels exclusively. The robot is small, all processing is performed on board, and the vision hardware consumes little energy. The overall robotic control is also simple, not requiring precise measurements or commands. The system is still able to achieve successful movement due to the reactive approach discussed above: if the world is sampled often enough, new commands can compensate for previous mistakes or imprecision.

By combining the vision modules, the system achieves good performance in multiple environments. In short, a collection of environment-specific modules is used to achieve environment-independence. This flexibility is unusual in vision systems, and it distinguishes this system in terms of performance, while other attributes, including reactive planning, behavior-based architecture, and computational simplicity distinguish it in terms of design. The next chapter discusses the system's performance.

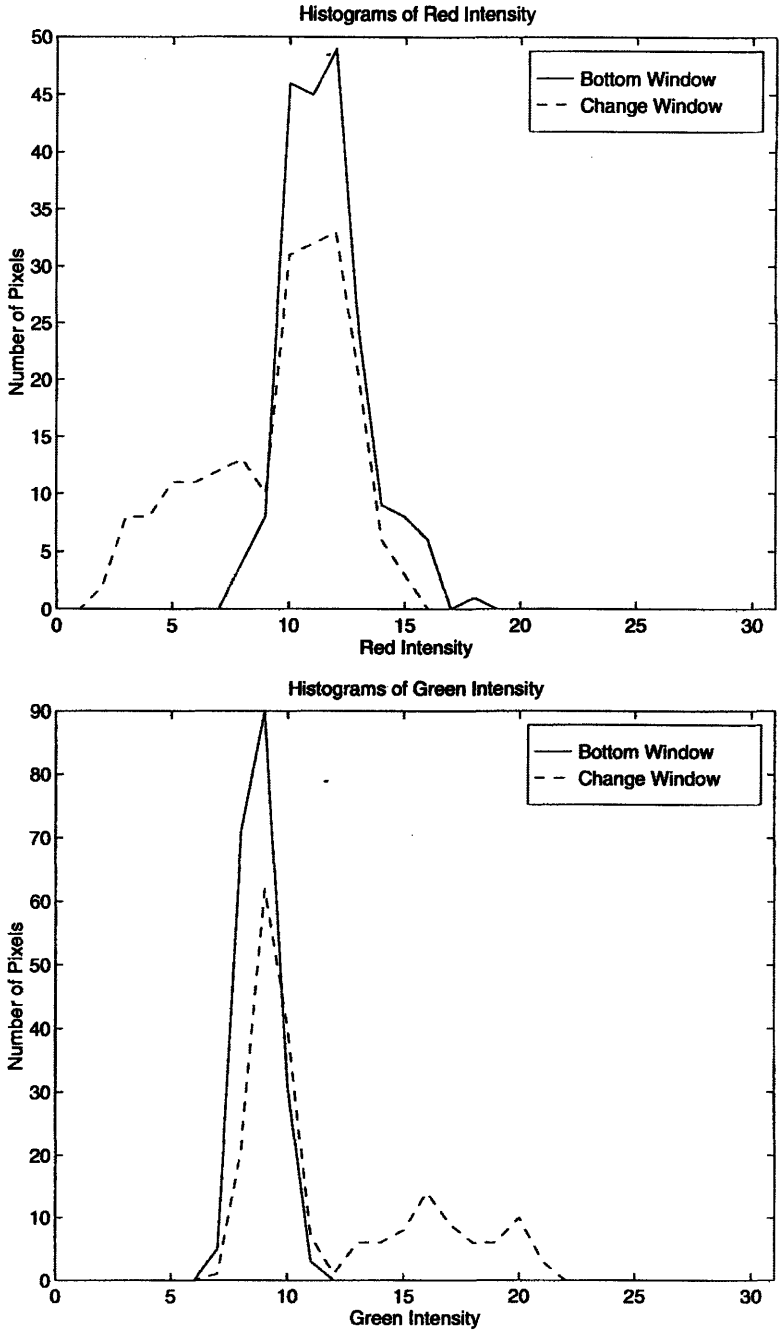

Figure 3-7: These images give the histograms of the normalized red and green components of the windows indicated by the above image. For the red histograms notice that the higher window contains fewer pixels of a medium red intensity and more pixels of a very small red intensity. This difference corresponds to the bright green pixels in the higher window which have a much smaller red component than the grey pixels that dominate the bottom window. The opposite difference is observed in the green histograms. For the higher window, the bright green pixels cause a "bump" in the histogram corresponding to large green components, while the number of pixels with medium green components is reduced.

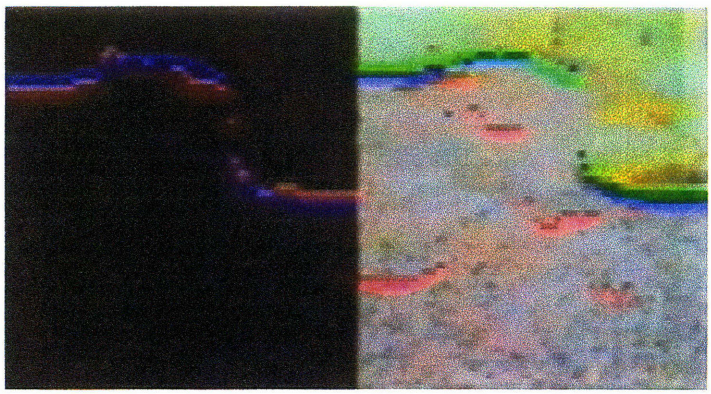

Figure 3-8: The right half of this image shows the depth arrays given by the three individual modules, overlaid on the image frame from which they were generated. The output of the brightness gradient magnitude ("edge") module is shown in red, the output of the normalized RGB module is shown in green, and output of the normalized HSV module is shown in blue. Notice that the output of the edge module is incorrect since it does not correctly indicate the boundary of the safe ground. Even without knowledge of the scene, it would be an unlikely candidate for a good boundary since it is jagged. The other two modules find an adequate boundary. The left half of this image shows the outputs of the two methods used to combine the modules: red shows the smoothness approach and blue shows the median approach.

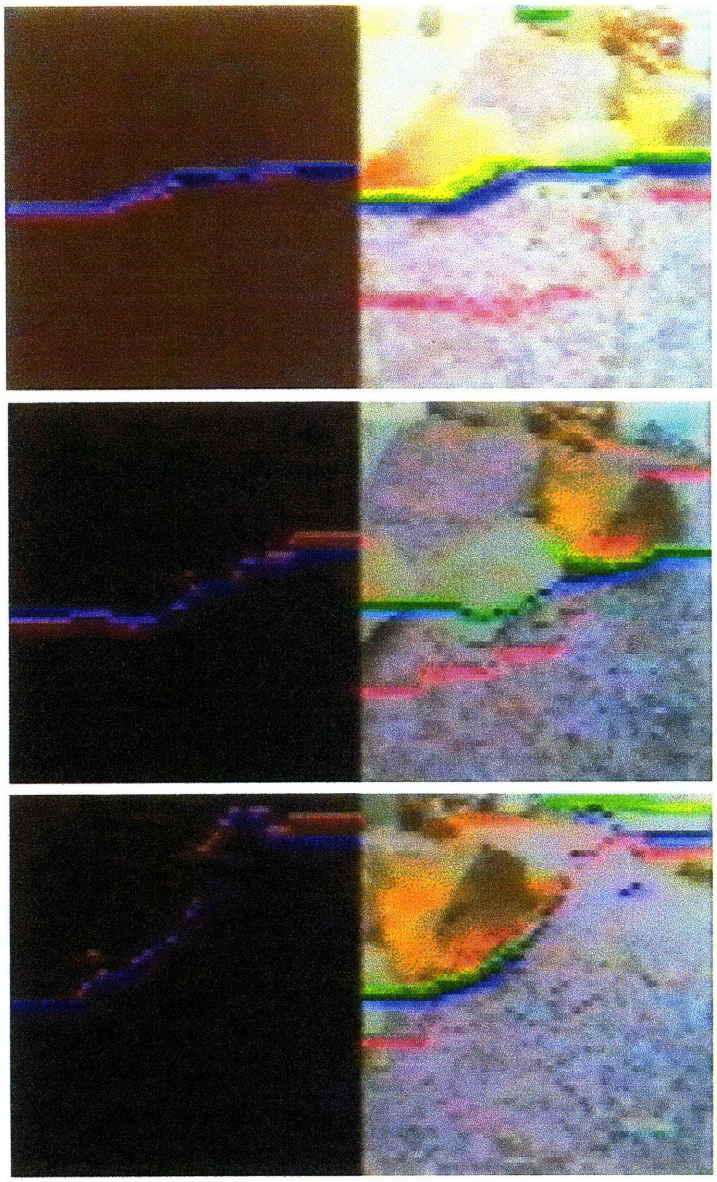

Figure 3-9: These images demonstrate the output of the the system on sequential image frames taken by the robot. In the first image, rocks are blocking the path of the robot. Since the right half of the obstacle boundary is slightly higher than the left half, the robot turns to the right. That difference in height is then magnified in the second image, so it turns more to the right. By the third image, it can see free space ahead on the right side, and needs only clear a single rock on the left side of the image, as detected. (Intermediate images are not shown.) The left half of each image shows the obstacle boundaries found by the two combining techniques: smoothness in red and median in blue, and the right half shows the boundaries given by the individual modules: brightness gradient magnitude in red, normalized RGB in green, and normalized HSV in blue.

Chapter 4

Results

The system has been run cumulatively for over 100 hours and demonstrated for over 200 visitors to the MIT AI Laboratory, including over 100 researchers from a major computer vision conference. Throughout development, these demonstrations varied in the modules that were included, the motor control mechanism, and other aspects of the system. For the most part, the results discussed here are those of the final system, although some discussion of the performance of individual modules is included. Other avenues explored in this work are discussed in section 4.4.

During experiments and demonstrations, when possible, images were received from a video transmitter on the robot Pebbles for purposes of display and understanding, overlaying the determined depth array onto the current image. Examples of the images that were displayed are shown in Figure 4-1.

Throughout this chapter, two locations are mentioned repeatedly as common testing sites of the system. The first site, the "sandbox," is a 15ft x 10ft simulated Martian surface, pictured in Figure 4-2. The ground is covered with coarse gravel of various shades of gray, and larger rocks are the obstacles. The ground is not a smooth plane, but rough throughout and also sloping in places. The second site, called "8AI", is a large lounge area of the the 8th floor of the MIT AI Laboratory. Pictured in Figure 4-3, this location is covered with a carpet of varying shades of orange. Obstacles include walls, two off-white sofas, bookcases, tables, and various chairs. Both of these locations are cluttered with obstacles. Furthermore, during testing in all locations,

people step in front of the robot and place other objects in its path.

4.1 Performance

In the testing of the system, success was measured by observing both the robot's behavior and the image obstacle boundaries (depth maps) found according to individual frames. Successful robot behavior is defined as traveling around a cluttered environment, moving straight forward when the path ahead is clear and turning around obstacles when encountered. The obstacle boundaries overlaid on original images are transmitted from the robot to a video monitor for verification that the success of the system is not due to noise or random behavior of the motors. Even without such output, however, it would be nearly impossible for randomness to account for successful obstacle avoidance, given the large amount of time and various environments in which the system has been tested.

The individual modules were tested separately before the testing of the entire system. Some interesting results were observed from tests of the edge module and the HSV module in 8AI. For obstacles such as the walls, the sofas, a large box, and people, neither module had any difficulty. However, the edge module occasionally missed a smooth bookcase with similar visual texture as the floor. But the HSV module had no trouble detecting the bookcase since it was a deep brown, rather different in color from the orange carpet. The opposite situation occurred in the case of a particular style of chairs found in 8AI. The legs of these chairs, pictured in Figure 4-4, are thin metallic bars that reflect the color of the carpet. Thus, the HSV module did not detect these legs.

These chairs are actually an unfair test of the system since they do not satisfy the environmental constraints utilized by the system. First, the bulk of the chair most easily detectable does *not* rest on the ground, and, more importantly, the obstacle boundary defined by these legs is only smooth in the case that the long edge of the leg is imaged horizontally across the image. However, with the long edge visible, even somewhat foreshortened, the edge module succeeded. This success is due to the fact

that the long edges of the chair legs created large gradient magnitudes which were different from the magnitudes computed over the carpet portion of the image. Both of these types of obstacles are correctly detected by the final system in which the modules are used together.

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 is only slightly larger than the robot's width. Pebbles successfully avoided most obstacles except for rocks too close together (Pebbles has no sense of its own width. This problem is discussed below.) and rocks outside the camera's field-of-view due to an unfortunate positioning after turning (This was rare and is also discussed below.). During the many hours of experiments in this environment, no effort was made to control the lighting conditions, and, as a result, they often varied. Performance was not harmed by these variations. When a person stood directly in the way of the robot, the robot either immediately assumed it was stuck and turned in place, or if some free space was visible it turned slightly back and forth until the person moved and it could proceed safely. In general, the robot can wander around the sandbox safely for large amounts of time, limited primarily by the life of the battery and occasionally by the failure modes mentioned below.

Another interesting behavior occurred with regards to a small region of sand within the room, pictured in Figure 4-5. When there was a sharp boundary between the gravel and the sand, the sand was perceived as unsafe since it was not known to be safe according to the constraint that the robot should stay on the same type of ground on which it starts. However, when gravel and sand were mixed between the the regions so the boundary was unclear, the robot occasionally proceeded onto the sand since the gradual change implied that the boundary would be detected higher in the image, so the robot would proceed forward. Upon proceeding forward, the bottom portion of the new image could become a mixture, and so on, until the robot was entirely in the sandy region. This is not a problem since if a visual change occurs very slowly it is unlikely that it is an obstacle, in natural environments.

Furthermore, the system succeeded in 8AI. In general, it roamed safely for long

periods of time provided the described constraints were met. Walls, sofas, boxes, chairs, and people were successfully avoided. Many people walked in front of Pebbles, including one spirited lab member[1] who jumped in front of it and lay down on the carpet, as pictured in Figure 4-7. Corridor following, even at corners, was easily accomplished by the system. Again, moderate lighting variations did not affect the system.

One interesting behavior of the robot in the 8AI environment occurs where the carpet is extended from the floor up onto the wall of a short bench in the room, pictured in Figure 4-6. Since no visual texture or color change occurs at this obstacle, the robot does not detect it. This miss, however, is not a failure of the system, but rather, it demonstrates that the system is specialized to operate in environments satisfying given constraints. When these constraints fail, success is not ensured.

It is interesting to notice that, when avoiding a nearby rock, Pebbles appears to be moving to improve its viewpoint, according to the strategy of active vision. The first image in Figure 4-1 shows its view of a rock, when Pebbles would turn only slightly. However, that slight turn makes the rock appear more prominently on one side or the other (the second image), so a sharper turn is made. The third image shows the process continuing until the obstacle is nearly cleared.

The robot was also tested in other rooms at the AI Laboratory. Variations included the type and amount of clutter and the pattern of the carpet. Preliminary testing was also performed outdoors. Some pictures of Pebbles operating in various environments are given in Figures 4-7 and 4-8.

4.2 Failure Modes

The system fails if obstacles are outside the field-of-view of the camera. This failure mode is not a shortcoming of the visual processing, but rather of the hardware configuration of the camera and the robot. On the current platform, this happens when a small obstacle, such as a small rock, is directly in front of one of Pebbles' treads, as

[1] Rene Schaad

pictured in Figure 4-9. If Pebbles had approached this rock by moving directly toward it, this configuration would be unlikely to occur since the rock would be detected and Pebbles would turn away from it. The problem can occur when Pebbles has been turning. Since no memory of past images is stored, this configuration could result when Pebbles has turned away from this rock just long enough for it to disappear from its field-of-view, but not long enough for it to be actually out of the way. As mentioned above, this problem is addressed by exaggerating turning motions when obstacles are nearby. Namely, the coefficients on the *turnangle* parameter are set to be larger than otherwise necessary. The configuration can also arise when Pebbles has just turned sharply away from a different obstacle, and the troublesome rock has never been detected. This situation requires that the obstacles be quite close together and is not addressed by the current system. Possible solutions are discussed in section 5.1.

A related failure mode is also due to Pebbles' limited field-of-view and lack of knowledge about its physical size. If two large obstacles are approximately equidistant from Pebbles, one on either side of the image, and there is a fairly wide space between them, Pebbles may try to go between them – even if the space is actually smaller than the width of Pebbles. When encountering the first image in Figure 4-10, noise leads to an arbitrary decision of which way to turn, say right. Subsequently, the second image in Figure 4-10 occurs, so Pebbles turns left. The third image of Figure 4-10 is then encountered, so Pebbles turns right. This process continues. If the overall obstacle boundary is high enough in the image, as is common in this case since the passageway is contributing a high segment to the boundary, Pebbles also moves forward, edging left and forward, then right and forward, until stuck.

A straightforward solution is to hardcode an estimation of the width of the robot, measured in image pixels. This estimation would be used when images like the first one in Figure 4-10 are encountered. If the passageway is too narrow, the robot should treat the multiple obstacles as one large obstacle. Note that the passageway must be almost as large as the robot for this problem to occur. When it is small, there is a smaller forward motion of the robot, and the robot turns more in place. The

exaggerated turn-angles direct the robot to turn away from the one rock enough that the other rock is usually cleared as well, or at least partially cleared so that subsequent obstacle boundaries will direct the robot to turn away from it completely.

Situations plagued by very bright direct lighting can cause the system to fail. During an outdoor trial, the bright sunlight reflected on the ground caused a "white-out" region in the image, as shown in Figure 4-11. In such situations, the boundary between the white-out region and the accurately imaged region is perceived as the obstacle boundary if it is lower in the image than any accurately imaged obstacle. Another lighting-related problem occurred indoors on a shiny linoleum floor as demonstrated in the second image of Figure 4-11. Bright specularities appeared on the floor and were perceived as safe ground. The accurately imaged blue floor was then perceived as an obstacle. Although the edge-based module correctly handled the scene, it was outnumbered by the color modules, which failed to find the true boundary. The most straightforward solution to the problems in these two situations is to use modules that are well-suited to such specularities, such as edge-based modules. Alternatively, the system could have knowledge of such reflections and discount them as obstacles, or it could automatically detect that an image was inadequate from this standpoint and stop or look for a better viewpoint. A hardware solution might be to modify the camera to handle these lighting conditions. Sharp shadows also pose a problem in bright sunlight in that the robot's own shadow may appear to be an obstacle, as shown in Figure 4-11.

Another problem occurs when the visual texture of the safe ground is very irregular or larger than the preset window size. This was seen in an outdoor experiment with sparse brightly colored fallen leaves, as pictured in Figure 4-12. Since the camera is positioned relatively low to the ground, these leaves appeared too large to be adequately modeled by a window of size 20 x 10. Compared to the window size, they are large enough to contribute significantly to the measures of any window they occupy. These measures will likely be different than measures of windows without the leaves, thus the leaves will wrongly indicate obstacles. Possible software solutions are to dynamically change the window size based on the perceived input or to use a

depth-from motion technique over multiple frames, and a possible hardware solution is to raise the camera so that larger textures appear smaller in the image.

4.3 Robustness Issues

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 vary by about 15 percent and still achieve good results. Furthermore, different image resolutions and window sizes have been tested with good results. For example, the initial system used a resolution of 220 x 110 and a 20 x 20 window. No change was required to operate the system at 128 x 64 and a 20 x 10 window other than modifying the thresholds corresponding to the necessary changes in histograms for a window to be classified as an obstacle. The resolution was then reduced to the current resolution of 64 x 64 with no change in window size. In addition to the speed-up in processing, the results were actually improved because windows became wider with respect to the overall image size. This allowed for a more stable estimation of the measure of the safe regions in the images. Thus, this code should be easily portable to various platforms and run with various resolutions.

The system withstands a variety of lighting changes, and other environmental variations. Such variations include changes in floor color, visual texture, and physical texture. Also, the amount and type of obstacles was varied considerably throughout the experiments.

4.4 Other Experiments

Other vision-based obstacle detection modules have been considered in addition to those chosen for the final system. Such modules are discussed here since they are appropriate for other environments. Lessons learned from varying the camera's position on the robot are also mentioned.

4.4.1 Depth From Motion

The constraints underlying the framework of the modules in the final system include the assumption that obstacles are visually detectable in single image frames. The system fails when visual boundaries in individual frames are not equivalent to obstacles, such as in the situations pictured in Figures 4-6 and 4-12. In the case of the leaves, the robot avoids safe regions and may determine that no safe path exists at all. In this case the constraint is too strict. In the case of the carpet-covered bench it is too lax. In such situations obstacles are invisible in single frames. Image motion over multiple frames is required to recover scene depth. Alternately, stereo could be used but is not considered for this work which explores the possibilities for a relatively simple monocular platform.

Much work has been done on recovering scene depth from sequences of images recorded with a moving camera. This research considered approaches that use all of the image intensities in the image instead of first detecting feature points such as corners from which the geometry could be reconstructed because the target unstructured environments lack such features. Specifically, approaches based on optical flow as discussed in 2.3.4 were considered.

Optical Flow

The initial approach of this work was to use an optical flow-based strategy for obstacle detection. Flow computation which is not correlation-based requires a large number of iterations for convergence, so a simplification derived in (Horn & Weldon 1988) to yield the depth directly without first determining the flow was tested. The assumption of pure forward translation on flat ground yielded the equation

$$Z = -(xE_x + yE_y)/E_t$$

where Z is scene depth, x and y are image coordinates, and E_x, E_y, and E_t are horizontal, vertical, and temporal image derivatives, respectively. This approach failed on the current platform because the robot motion was not completely controlled.

The motion of the motors and the noise of the framegrabber caused instability in the image even in experiments on a flat carpeted floor. Moreover, the robot's motion was not actually restricted to pure forward translation but small rotations were allowed as they could be used to avoid obstacles. The rotations and noise invalidated the simplification. A possibility for future work is to incorporate rotation into the direct flow equation, as also described in Horn & Weldon (1988).

Divergent Stereo, Normal Flow

A module based on the divergent stereo experiments in (Santos-Victor et al. 1995) was implemented. For a moving system, there is more optical flow in an image region that corresponds to near objects than one which corresponds to distant objects. Divergent stereo was approximated on a monocular system by splitting the image into halves. Then the robot moves in the direction of the image half with less total flow. More generally, one does not need to explicitly divide the image in half but can just move away from regions of large flow since the goal is to balance the flow across the two halves of the image.

This approach had difficulty perhaps partially due to insufficient flow information since the camera was not performing actual divergent stereo. In the case of (Santos-Victor et al. 1995), two cameras were used, one facing directly left and one facing directly right. Obstacles were restricted to those that are avoided by the robot's always centering itself between the walls on either side of its path, where a path is usually a corridor. This centering behavior is achieved by balancing the flow across both images. With the camera moving laterally with respect to the walls or obstacles, the situation is optimal for detecting normal flow, as the entire magnitude of the flow is represented by this normal component. In the current set-up with obstacles directly in the path of the single camera at the front of the robot, most of the motion is not restricted to the normal component of the optical flow so more precision is required in the detection of changes in normal flow as they are smaller. Complicated by rough ground, a noisy framegrabber, and low resolution, the current situation may not be well-suited to this technique.

It is likely that related work involving a more general use of normal flow will be more suitable for this platform (Santos-Victor & Sandini 1995). The related system is monocular and requires no camera calibration. The environment is constrained to have an approximately flat ground. The system operates by initially computing the expected normal flow in the image of a flat ground plane with no obstacles. This flow depends on the camera position, tilt, geometry, and speed. For obstacle detection, normal flow is computed as the camera moves. This new flow is compared with the initial flow, and image regions where the two flows differ are considered obstacles since obstacles lie outside the ground plane. Work-in-progress for this system includes experimenting with this strategy, using it for the detection of slope changes in the terrain in addition to obstacle detection.

4.4.2 Histogram Measures

Each of the modules incorporated into the current system uses a measure that is the histogram of a particular image property over a fixed size window. The properties included are brightness gradient magnitude, normalized RGB color, and normalized HSV color. This framework can be extended to incorporate any property given by a function that is computed locally at each image pixel and has a fixed range.

The histogram framework has been used with a wide variety of such functions including thresholded brightness gradients, greyscale intensity, unnormalized RGB color, RGB gradient magnitudes, and edge orientations. Both negative and positive results were observed.

Thresholded brightness gradients, where the function output was 0 or 1 corresponding to "edge" or "no edge", failed for unstructured environments since the results were too dependent on the threshold. For given images, a particular threshold would be appropriate to give fair results but even within the same region of the environment, for example, within the gravel region of the sandbox, slight shadows and lighting variations necessitated different thresholds. A possible solution is an adaptive threshold, as used (but for a different purpose) in (Horswill 1993*a*), but finding correct thresholds may be difficult as lighting inconsistencies are often present even within

a single image. Although giving reasonable results, greyscale intensity performed strictly worse than color histograms, so it is not recommended if color information is available. Unnormalized RGB color was worse than its normalized analogue in performance since it was too sensitive to lighting inconsistencies, and also in efficiency since it required a triplet of histograms instead of a pair of histograms.

On the positive side, RGB gradient magnitudes were very effective in some environments and suggest the composition of modules as a method of generating additional modules. The edge orientation module was also successful. If computational power is available, one could incorporate these additional modules and others to achieve increased reliability through redundancy.

4.4.3 Camera Configurations

During system development, the camera position was changed with respect to the robot. Initially the camera was 4.5 inches above the ground and directed straight ahead. With this position, obstacles at varying distances from the robot appear at only slightly different heights in the image. Raising the camera to 10.5 inches and tilting it slightly down toward the ground increases this difference at the cost of compressing very close obstacles. The camera was repositioned accordingly and the overall estimation of depth using image height was much improved. For finding depth according to optical flow, however, it may be preferable to use the initial camera position.

Additionally, since raising the camera, the angle at which it is tilted toward the ground has varied. The system is robust to any tilt at which a reasonable field of view is achieved. Obstacles and the horizon appear higher in the image as the camera is tilted downward. This height directly determines the forward speed of the robot. For this reason, if the tilt is changed significantly and a particular motion speed is desired, the hardware-dependent coefficients used in the computation of the motor commands must be re-estimated.

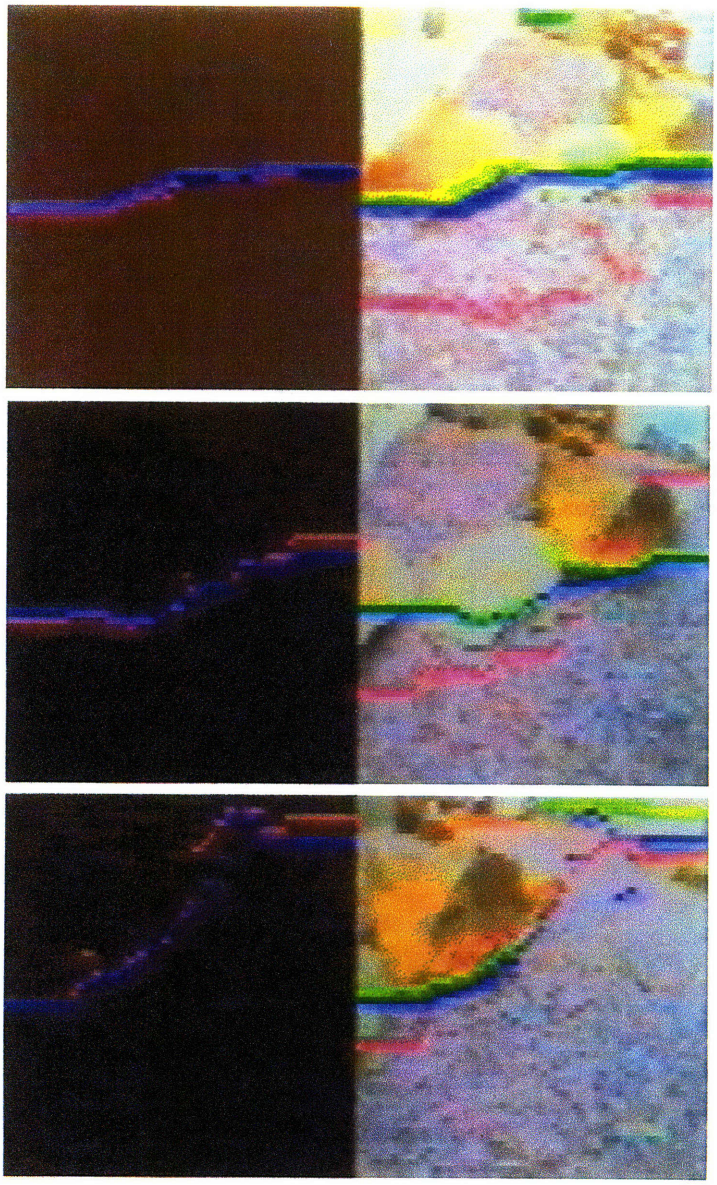

Figure 4-1: These images demonstrate the output of the the system on sequential image frames taken by the robot. In the first image, rocks are blocking the path of the robot. Since the right half of the obstacle boundary is slightly higher than the left half, the robot turns to the right. That difference in height is then magnified in the second image, so it turns more to the right. By the third image, it can see free space ahead on the right side, and needs only clear a single rock on the left side of the image, as detected. (Intermediate images are not shown.) The left half of each image shows the obstacle boundaries found by the two combining techniques: smoothness in red and median in blue, and the right half shows the boundaries given by the individual modules: brightness gradient magnitude in red, normalized RGB in green, and normalized HSV in blue.

Figure 4-2: The "sandbox", a simulated Martian surface on which the system was tested.

Figure 4-3: "8AI", a lounge area at the MIT AI Laboratory in which the system was tested.

Figure 4-4: Illustration of the advantage of multiple modules. Chairs of this variety yielded an interesting observation about the system, due to the design of the chair legs. Since the were metallic and very thin, they were often undetected by the color-based module. The edge-based module, however, had no trouble in this situations, as long as a reasonable portion of the long edge of the chair leg was visible. Since there were no sharp edges in the ground, the sharp edges defined by the chair leg yielded a very different histogram. Further, this example demonstrates the advantage of multiple modules, as the final system has no trouble with such situations.

Figure 4-5: This picture shows a region of the sandbox where the gravel ground mixed with a sandy region. When the boundary between the gravel and the sand is sharp, the robot does not cross the boundary; when the change is more gradual, the boundary is crossed.

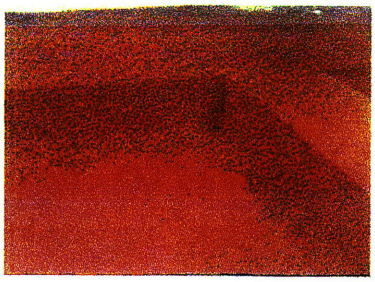

Figure 4-6: Since this carpet-covered bench exhibits no difference in visual cues from the safe ground, it fails to satisfy the a domain constraint used by the system. Thus, it is not successfully avoided.

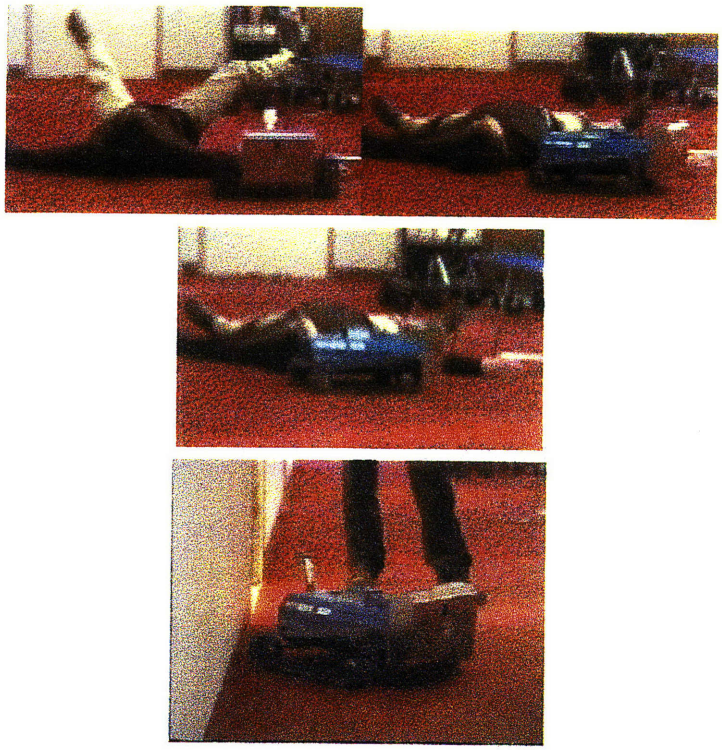

Figure 4-7: Testing Highlights. These images show the system operating in 8AI. The first three images are sequential and show the robot turning away from an obstacle.

Figure 4-8: More Highlights.

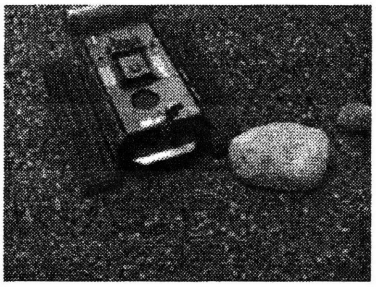

Figure 4-9: This image demonstrates the inadequacy of the field-of-view observed by Pebbles' camera. The camera is located at the front of the robot, at the top of the metal brace. The rock is an obstacle yet it is outside the view of the camera. In determining the motor commands from the visually-detected obstacle boundary, turn angles are exaggerated to make this configuration much less likely than otherwise. However, there remain situations in which it can occur.

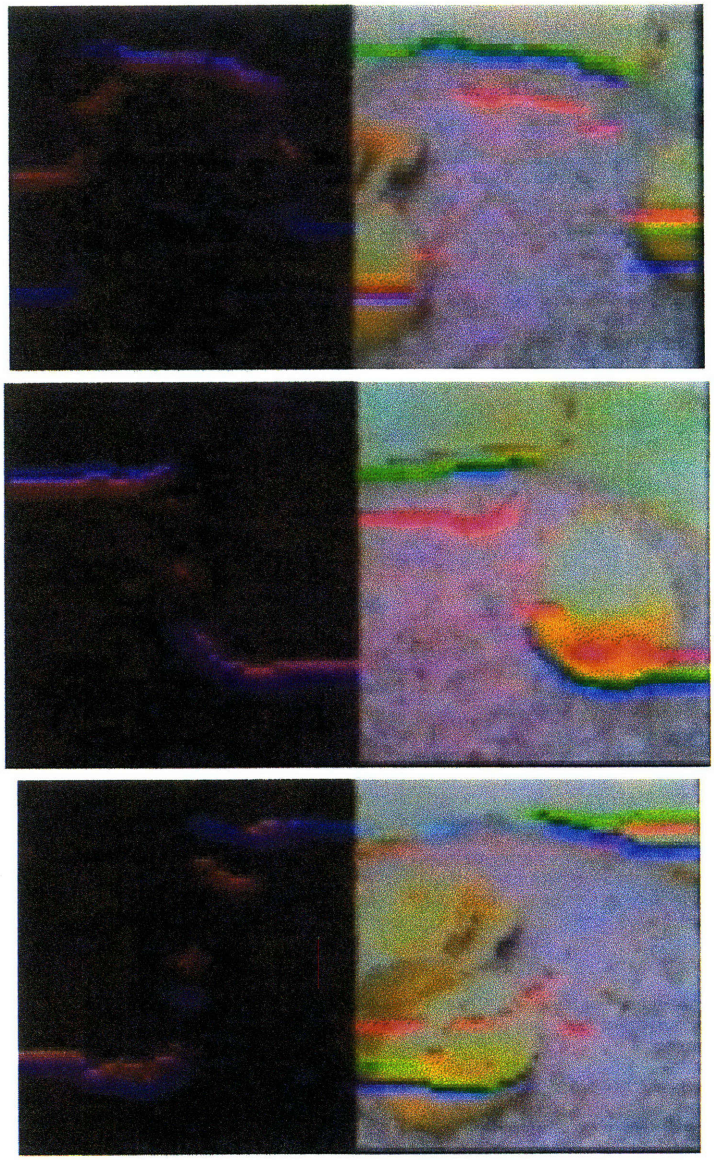

Figure 4-10: A failure mode. The first image illustrates a situation in which the space between the rocks on either side is slightly too small for the robot to traverse. Since Pebbles has no concept of its own width, and no memory of past obstacles, it treats this situation as follows. At the first image, the turn direction is almost arbitrary. In this example, the robot turns right. The rock to the left is now invisible to the robot (the second image), so it turns left to avoid the rock on the right. It then turns right again, in an attempt to avoid the rock on the left (the third image). This process continues. Since the free space ahead implies that the overall obstacle boundary is high enough to yield forward motion, the robot is also moving forward as it is turning. Thus, it eventually gets stuck on a rock. Note that this situation only occurs when the free path is almost (but not quite) wide enough to accommodate the robot. When the space is smaller, there is less of a forward component to the robots motion, and it turns more in place. Moreover, the exaggerated turn-angle coefficients often make the robot turn enough to avoid both rocks, or at least enough to perceive space on the far side of the second rock, so that subsequent images will complete the turn.

Figure 4-11: The first image, taken outdoors by the camera on the robot, demonstrates one trouble caused by sunlight. The white-out region of the image is the reflection of the sun on the ground and could be detected as an obstacle by the color modules of the system. A similar problem occurs indoors with shiny linoleum floors and bright ceiling lights, as pictured in the second image. In this scenario, the floor is blue, and the bright splotches are the reflections of the ceiling lights. The right half shows the depth arrays of the individual modules. Notice that the edge-based module, shown in red, gives the correct obstacle boundary, but the two color modules, pictured in green and blue, falsely determine that the specularities indicate safe ground and the blue color corresponds to obstacles. The color modules are not specialized to handle this environment, so they fail. A total system for this environment must ensure that the final output corresponds to modules, such as the edge-based module, that are specialized to this environment. The third image, taken outdoors by the robot's camera, demonstrates the other problem caused by bright sunlight: shadows. In this case, Pebbles is moving forward with a low sun at its back, and its shadow appears as an obstacle. (Notice the raised camera appearing tall in the shadow.)

Figure 4-12: Taken outdoors from the camera on Pebbles, this image of sparse brightly colored fallen leaves demonstrates visual texture that is too large and too irregular to be captured by the current window size of 20 x 10.

Chapter 5

Future Work

The current obstacle avoidance approach motivates several areas of future work. This chapter discusses engineering concerns raised by the current system and suggestions for higher-level navigation capabilities, dynamic module selection and other potential modules. It concludes with ideas about relaxing constraints in future systems.

5.1 Engineering Concerns

Pebbles' limited field-of-view caused problems for the system. The difficulty occurred when obstacles were in front of Pebbles' treads but outside the field-of-view of its camera. One option is to increase the field-of-view of the camera or to use other sensors for close-range obstacles, such as infrared, bump sensors, or additional cameras aimed at the blind-spots. A second option is to decrease the size of the robot. Horswill's Polly avoided this problem by using both solutions: a higher camera gave a wider view, and the robot base was smaller in comparison with the current system. One could also solve this problem in software by adding some memory of the depths of previously seen obstacles. At any given time, the depth map would extend outside the actual image to represent a larger virtual image. Motor commands would derive from this larger map. Future work could include such hardware or software modifications. Other possible software solutions could be explored where one must make decisions about what to represent, at what level of detail, and how to use this

information.

Another engineering issue concerns outdoor operation. Preliminary outdoor tests have revealed problems due to bright sunlight. Some trouble could be alleviated through the use of a special camera or lens filter. Since the white-out regions are not constant, or even linear, over an image, it would be desirable to have a non-linear filter. It also may be necessary to incorporate software solutions such as specialized knowledge-based detectors for specularities and shadows, including the robot's own shadow. One could consider the method of using expected color distributions for segmentation of roads as a possible motivation for a shadow detector (Turk et al. 1988). Following this work, it may be possible to hypothesize the expected color distribution of shadows.

5.2 Higher-Level Navigation

Some higher-level navigation capability is desirable in many robotic applications, even if it be user-assisted. Incorporating such additional modules into the current obstacle avoidance framework is straightforward although it is outside the scope of this thesis. Perhaps the most straightforward way to do this is to add additional layers implementing higher-level navigation according to the subsumption architecture (Brooks 1986). This navigation could be based on a variety of sensory modalities depending on the target, including vision, infrared, motor-based odometry, radio-positioning, and sonar. Alternatively, the current obstacle avoidance system could be incorporated into an interactive approach for directing a robot to a target. Although the high-level instructions would be user-specified, obstacle avoidance would be performed autonomously.

5.3 Module Selection

Another area of future work is in the run-time selection of modules. It may be desirable to run only a small subset of the modules on any given frame if processing

power is particularly limited in a given application or if a system includes a very large number of modules. Several module selection mechanisms are hypothesized.

One idea is to use the output or intermediate processing from one module to "excite" or "inhibit" another module. These rules would have to be designed specifically for the particular modules in a system. An example of this approach would be to use the intensity range of an image frame to excite or inhibit an edge-based module. The intensity range could be determined as a by-product of an intensity-based module. If this range is very small. edge information is probably minimal so the edge-based module should be ignored.

A similar but more general approach not specifically based on another module is to determine the dynamic range over an image frame of each of the bands of information utilized by each module. such as brightness gradient, intensity. red, green, blue, hue, saturation, and value. The module should be omitted if the range is very small as the band contains little information. If the range is significant. the module should be used and perhaps weighted heavily in the final determination of the motor commands.

To determine the reliability of a module, one could measure the persistence of the output of that module. If its depth array moves only slightly between image frames, it would receive a good persistence score. A module whose depth array changes drastically between images would receive a poor score. This persistence measure corresponds to the environmental constraint that if successive images are taken at small time intervals then the scene will not change very much: hence successive depth arrays should look similar. This persistence measure could be integrated over a short period of time to obtain an overall persistence value for a module in a current environment. Module selection could proceed by choosing only persistent modules. This persistence measure should be re-evaluated periodically since even a particular environment may change over time.

5.4 Potential Modules

The current approach is extensible to systems appropriate for any combination of environments such that at least some vision module works in each domain. For future systems, new modules can be designed in accordance with the environments in which the systems are to operate.

Some potential modules not yet implemented are modules based on frequency analysis or a neural network in which training would occur in the desired environments. Another possibility is a module to detect the change in slope of the ground ahead of the robot, probably using a depth-from-motion technique. The approach currently being considered for use with the system is to compute the normal flow of a flat ground plane then to compute the actual normal flow of the scene (Santos-Victor & Sandini 1995). Differences between the flow patterns indicate differences in depth and could be used to detect obstacles or changes in overall slope of the ground plane.

5.5 Relaxing Constraints

Both hardware and software restrictions were imposed on the scope of this research. Future work could explore the implications of relaxing these constraints. The amount of processing was limited by the requirement that all processing proceed on-board the robot. Also, a single camera image was the only sensory input. Future work could increase processing power and incorporate additional cameras or other sensors. Such sensors could support independent processing modules or operate in conjunction with the existing camera. It is straightforward to incorporate new modules into the existing framework. Software constraints included storing little or no information about previous images. Future research could explore the option of using similar vision modules in a less reactive system.

Chapter 6

Conclusions

This thesis describes an autonomous visually-guided obstacle avoidance system on a mobile robot designed and implemented according to the following outline. 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 texture or color information. Each module receives the image directly from the camera and returns a one-dimensional depth array indicating the perceived locations of obstacles. These arrays are combined into a single array according to preferences determined by the contents of the arrays themselves. Motor commands are generated from this array.

The objective of using several specialized modules is to achieve environment-independence through multiple environment-specific modules. The modules are designed according to observed domain constraints, such as restrictions that all objects rest on the ground and that the ground is approximately flat. Further, the vision system stores no information across image frames and operates at very low image resolution on a small monocular robot with all processing performed on-board. The approach of combining multiple computationally simple modules into a system for robustness across widely varying environments can also be applied to sensors other than a single camera. The framework described in this work can be extended to include multiple cameras, other sensors, or a combination of various sensors.

Throughout development, the system has been tested cumulatively for over 100

hours and successfully avoids obstacles in a variety of environments including indoor carpeted rooms and a simulated Martian surface. Obstacles include rocks, furniture, walls, and people. In these domains, failures are usually due to an insufficient field of view of the camera or to the system's lack of knowledge of the robot's width. Preliminary outdoor trials suggest that knowledge about shadow and specularities should be incorporated into the system if it is intended for use in bright sunlight.

Bibliography

Agre. P. E. & Chapman, D. (1987), Pengi: An implementation of a theory of activity, *in* 'Proceedings of the Sixth Nat'l Conf. on Artificial Intelligence', pp. 268–272.

Aloimonos. Y. (1993), *Active Perception*, Lawrence Erlbaum Assoc.

Ballard. D. H. (1991), 'Animate vision'. *Artificial Intelligence* **48**, 57–86.

Beer. R. (1988), A dynamical systems perspective on autonomous agents. *in* 'CES 92-11. Case Western Reserve University'.

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

Brooks. R. A. (1994), *L*, IS Robotics, Inc.

Coombs. D. (1992), Real-Time Gaze Holding in Binocular Robot Vision. Technical Report Rocherster TR 415, University of Rochester, Computer Science.

Coombs. D. & Roberts, K. (1992), "Bee-bot": using peripheral optical flow to avoid obstacles, *in* 'Intelligent Robots and Computer Vision', Vol. 1825, SPIE. pp. 714–721.

Crisman. J. D. (1991), Color Region Tracking for Vehicle Guidance, *in* 'Active Vision', The MIT Press, Cambridge, MA, pp. 1080–1085.

Dickmanns. E. D., Mysliwetz, B. & 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.

Franceschini, N., Pichon, J. & Blanes, C. (1991). Real time visuomotor control: from flies to robots. *in* 'Fifth Int'l Conf. on Advanced Robotics', Pisa, Italy.

Gavin, A. S. (1994). *Low Computation Vision-Based Navigation For Mobile Robots*, Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Gomi, T. (1995*a*). Autonomous Wheelchair Project. Applied AI Systems, Inc., Kanata, Ontario, Canada.

Gomi, T. (1995*b*), A Highly Efficient Vision System for Fast Robot/Vehicle Navigation, *in* 'Proceedings, Second Asian Conf. on Computer Vision', Singapore.

Gomi, T., Ide, K. & Maheral, P. (1995), Vision-based Navigation for an Office Messenger Robot, *in* 'Volker Graefe, Elsevier Science, editor, Intelligent Robots and Systems'.

Gomi, T., Ide, K. & Matsuo, H. (1994), The Development of a Fully Autonmous Ground Vehicle (FAGV), *in* 'Intelligent Vehicles Symposium', Paris, France.

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.

Horn, B. K. P. (1986). *Robot Vision*, MIT Press, Cambridge, MA.

Horn, B. K. P. & Weldon, E. J. (1988), 'Direct Methods for Recovering Motion', *Int'l Journal of Computer Vision* **2**, 51–76.

Horridge, G. A. (1986), A theory of insect vision: velocity parallax, *in* 'Proceedings of the Royal Society of London, B', Vol. 229, pp. 13–27.

Horswill, I. D. (1993*a*). Polly: A vision based Artificial Agent, *in* 'Proceedings of the Eleventh Nat'l Conf. on AI', AAAI, MIT Press, pp. 824–829.

Horswill, I. D. (1993*b*), *Specialization of Perceptual Processes*, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Horswill, I. D. (1995), 'Visual routines and visual search: a real-time implementation and an automata-theoretic analysis', *submitted to IJCAI.*

IS Robotics, I. (1995), *The Pebbles III Robot, User Manual, Version 2.0.*

Krotkov, E. & Hebert, M. (1995), Mapping and Positioning for a Prototype Lunar Rover, *in* 'Proceedings of IEEE Int'l Conf. on Robotics and Automation', pp. 2913–2919.

Krotkov, E. P. (1989), *Active Computer Vision by Cooperative Focus and Stereo*, Springer-Verlag New York, Inc.

Langer, D., Rosenblatt, J. K. & Hebert, M. (1994), 'A Behavior-Based System for Off-Road Navigation', *IEEE Transactions on Robotics and Automation* **10**(6), 776–783.

Lynn, S. W. (1994), A Motion Vision System for a Martian Micro-Rover, *in* 'Draper Laboratory CSDL-T-1197', Master's Thesis, Massachusetts Institute of Technology.

Mahoney, J. V. (1993), Elements of signal Geometry, Technical report, XEROX Palo Alto Research Center.

Mahoney, J. V. (forthcoming), Signal-based figure/ground separation, Technical report, XEROX Palo Alto Research Center.

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', Vol. 1, pp. 433–440.

Matthies, L., Szeliski, R. & Kanade, T. (1988), Kalman Filter-based Algorithms for Estimating Depth from Image Sequences, *in* 'Carnegie Mellon University, The Robotics Institute, Technical Report CMU-RI-TR-88-1'.

Pagnot, R. & Grandjean, P. (1995), Fast Cross-Country Navigation on Fair Terrains, *in* 'Proceedings of IEEE Int'l Conf. on Robotics and Automation', pp. 2593–2598.

Pahlavan, K. & Eklundh, J. (1992), 'A Head-Eye System - Analysis and Design', *CVGIP: Image Understanding* **56**, 41–56.

Payton, D., Rosenblatt, J. & Kersey, D. (1990), 'Plan Guided Reaction', *IEEE Transactions on Systems, Man, and Cybernetics* **1**(6), 1370–1382.

Rosenblatt, J. & Payton, D. (1989), A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control, *in* 'Proceedings of IEEE Int'l Joint Conf. on Neural Networks', Vol. 2, pp. 317–323.

Rosenblatt, J. & Thorpe, C. (1995), Combining Multiple Goals in a Behavior-Based Architecture, *in* 'Proceedings of IEEE Int'l Conf. on Intelligent Robots and Systems', Vol. 1, pp. 136–141.

Santos-Victor, J. & Sandini, G. (1995), Uncalibrated Obstacle Detection using Normal Flow.

Santos-Victor, J., Sandini, G., Curotto, F. & Garibaldi, S. (1995), 'Divergent Stereo in Autonomous Navigation: From Bees to Robots', *Int'l Journal of Computer Vision* pp. 159–177.

Simmons, R., Krotkov, E., Chrisman, L., Cozman, F., Goodwin, R., Hebert, M., Katragadda, L., Koenig, S., Krishnaswamy, G., Shinoda, Y., Whittaker, W. & Klarer, P. (1995), 'Experience with Rover Navigation for Lunar-Like Terrains', *Proceedings of IEEE Int'l Conf. on Intelligent Robots and Systems* **1**, 441–446.

Smithers, T. (1995), What the Dynamics of Adaptive Behavior and Cognition Might Look Like in Agent-Environment Interaction Systems, *in* 'Practice and Future of Autonomous Agents', Vol. 2, pp. 0–27.

Srinivasan, M. V., Lehrer, M., Kirchner, W. H. & Zhang, S. W. (1991), 'Range perception through apparent image speed in freely flying honeybees', *Visual Neuroscience* **6**, 519–535.

Turk, M. A., Morgenthaler. D. G., Gremban, K. D. & Marra. M. (1988), 'VITS - A Vision System for Autonomous Land Vehicle Navigation', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**(3), 342–361.

Ullman, S. (1984), 'Visual Routines', *Cognition* **18**, 97–159.

Zeng, N. & Crisman. J. D. (1995), Categorical Color Projection for Robot Road Following, *in* 'Proceedings of IEEE Int'l Conf. on Robotics and Automation', IEEE, pp. 1080–1085.