A Semi-Autonomous Vision-Based Navigation System for a Mobile Robotic Vehicle

by

Edward Y. C. Huang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2003

© Edward Y. C. Huang, MMIII. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

Author . . . .
Department of Electrical Engineering and Computer Science
May 21, 2003

Certified by . . . .
Kenneth Houston
Charles Stark Draper Laboratory
Thesis Supervisor

Certified by . . . .
Prof. Seth Teller
Associate Professor of Computer Science and Engineering
Thesis Advisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

A Semi-Autonomous Vision-Based Navigation System for a Mobile Robotic Vehicle

by

Edward Y. C. Huang

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Direct teleoperation of a mobile vehicle over a communications channel with delay on the order of one second is problematic for precise maneuvering and obstacle avoidance. To curb both operator vigilance and maneuvering mishaps during vehicle teleoperation, this thesis aims to develop a semi-autonomous vision-based navigation system for a small mobile robotic vehicle designed by Draper Laboratory. The system relies on monocular vision processing to employ an optical flow balance strategy for real-time obstacle avoidance. The system utilizes a multimedia processor for video processing and is implemented within a payload module built for a visual servoing task. The system design relies on a flexible communication protocol to integrate with the robot and operator control unit. Overall, the system is a useful augmentation for a flexible robotic platform for enabling better reconnaissance.

Thesis Supervisor: Kenneth Houston
Title: Senior Member of Technical Staff

Thesis Advisor: Prof. Seth Teller
Title: Associate Professor of Computer Science and Engineering

Acknowledgment

May 21, 2003

I wish to thank everyone at Draper Lab who helped and guided me as I worked as a summer VI-A intern and as a Draper Fellow working on my thesis. Special thanks go out to Scott Rasmussen, Ken Houston, Ed O'Neil, David Diel, and Rob Larsen.

I also wish to thank Professor Seth Teller for being a most helpful and understanding advisor for my thesis and for my entire course of study at MIT.

Finally, I wish to express my thanks to all of my family and friends for their love and support.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Internal Company Sponsored Research Project 15268 (IRD03-2-5042).

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

Edward Huang

Assignment

Draper Laboratory Report Number T-1440.

In consideration for the research opportunity and permission to prepare my thesis by and at The Charles Stark Draper Laboratory, Inc., I hereby assign my copyright of the thesis to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts.

Edward Huang Date

[This page intentionally left blank]

Contents

1 Introduction 17

1.1 Overview 17

1.2 Motivation 20

1.3 Road Map 20

2 Background and Related Work 23

2.1 Reconnaissance Robots 23

2.1.1 Unmanned Aerial Vehicles 24

2.1.2 Ground Reconnaissance Robots 24

2.1.3 Operator Control Schemes 25

2.2 Vision for Mobile Robot Navigation 26

2.2.1 Domain Specialization 27

2.2.2 Obstacle Avoidance 27

2.2.3 Visual Servoing 29

2.3 Robot System Architecture Overview 29

2.3.1 Mechanical 31

2.3.2 Electronic Hardware 31

2.3.3 Sensor Suite 32

2.3.4 Operator Interface 33

3 Navigation System 35

3.1 Optical Flow Obstacle Avoidance 37

3.1.1 Defining Optical Flow 37

3.1.2 Calculating Optical Flow 37
3.1.3 Determining Optical Flow by Patch Matching 38
3.2 Optical Flow Balance Strategy 39
3.3 Optical Flow Optimizations 40
3.3.1 Searching for the Best Patch Match 40
3.3.2 Domain Specific Optimizations 42
3.3.3 Other Potential Improvements 43
3.4 Robot and OCU Communication Protocol 43
3.4.1 Payload Module Control 45
3.5 Implementation Details and Optimizations 46
3.5.1 Imbalance Thresholding and Translating Flow Imbalance to Drive Commands 46
3.5.2 Image Noise Reduction 47
3.5.3 Time Averaging Flow 49
3.5.4 Program Timing Constraints 51
3.6 Supervisory Control of Robot 52

4 Testing and Results **55**
4.1 Simulated Environment 55
4.1.1 Setup 55
4.1.2 Results 56
4.2 Real World Testing 60
4.2.1 Setup 60
4.2.2 Results 60

5 Future Work **67**
5.1 Improvements to Current Optical Flow Obstacle Avoidance 67
5.2 Utilizing Full Robot Suite 68
5.3 Adding New Behaviors 69
5.4 Building Better Environment Models 69

6 Conclusion **71**

List of Figures

1-1 Draper Laboratory's High Mobility Tactical Microrobot. The robot is completely controlled by an operator using the IPAQ personal digital assistant (PDA) shown. 19

2-1 Block Diagram of Robot Architecture. 30

2-2 An active Operator Control Unit (OCU). 33

3-1 The robot with payload module. 36

3-2 Packets used in the OCU/robot communication protocol. 44

3-3 Image pair of a stationary scene and their noisy difference image. The top images show the same stationary scene acquired by a non-moving camera. The bottom image shows the difference of top image pair. The noisy non-zero difference image C presents a problem for accurate optical flow computation between the image pair. 48

3-4 Image pair of a scene with motion and their absolute difference image. The top two images show the subject moving his hand between the two frames. The bottom image shows the absolute difference of the image pair, with large magnitudes on the right side where the hand moved. In this case, the image difference imbalance is great enough to mask image noise. 50

3-5 Flowchart for deciding between operator and obstacle avoidance control. 53

4-1 Successive frames of the virtual robot going through an open doorway. 57

4-2 The virtual robot roaming through open rooms. The left frame shows the robot approaching the corner of the room with stone brick walls. The middle frame shows the mosaic pattern walls. The right picture shows the robot's field of view after colliding into the mosaic wall. Notice that all the optical flow vectors and their respective patch centers are visible. 58

4-3 Frame captures of the virtual robot roaming in a virtual environment. The map is laid out as two rooms joined by a long corridor. The top-left frame shows the virtual robot traveling down the hallway. The top-right frame shows the virtual robot following along the brick-textured wall. The rest of the frames show the virtual robot encountering various regularly shaped obstacles. All the frames are overlaid with all non-zero optical flow vectors, which are colored in black. Each flow vector emanates from the center of a patch used to compute optical flow. . . 59

4-4 The robot driving between a bucket and a table. The robot approaches the bucket (left figure), turns right to avoid the bucket, and drives between the bucket and the table leg (right figure). 61

4-5 The robot avoiding an obstacle in the middle of the lab. The robot approaches the cart from the side, as shown in the left figure, and the robot is able to avoid the obstacle, as shown in the right figure. . . . 61

4-6 The robot avoiding an office chair. The robot approaches a chair (top-left figure) and avoids it by turning left (top-right). The robot continues going forward until encountering the desk drawers, where it avoids it by turning left (bottom). The robot is now caught in a dead end under the desk. 62

4-7 The robot traveling down a hallway. The upper figure shows the robot trying to drive along the right wall after turning left to avoid collision, but the robot is too close to the wall, and thus friction is preventing forward progress. The bottom two figures show the robot unsuccessfully avoiding the left wall. The bottom-left figure shows the starting position of the robot, and the bottom-right figure shows the robot colliding into the wall. 64

4-8 The robot driving between a table and cabinet. On the left, the robot collides into the lightly textured cabinet. On the right, the robot is able to avoid the cabinet after an additional textured obstacle is placed in its way. 65

4-9 The robot driving through a doorway. When the edge of the doorway is centered in the robot's field of view, the robot is unsuccessful in avoiding it (top figure). When the robot drives forward with the edge of the doorway off to one side of its field of view (bottom-left), the robot is able to avoid it (bottom-right). In this case, the robot then is unable to avoid colliding into the door. 66

Chapter 1

Introduction

1.1 Overview

Reliable means of reconnaissance for gathering military information is necessary for successful military missions. Innovations in the tools used by reconnaissance agents have thus continuously been engineered. These innovative tools include machines that enable soldiers to gather reconnaissance information without human ventures into hostile territory. Current tools being used include the Predator and Global Hawk unmanned aerial vehicles (UAV), which are used to gather aerial photographic reconnaissance, and the Packbot mobile robotic vehicle, which is used for ground reconnaissance in urban environments.

This thesis focuses on a semi-autonomous navigation system for the High-Mobility Tactical Micro-Robot (HMTM), a small mobile robotic vehicle being designed at Draper Laboratory (Figure 1-1). The robot is designed for both indoor and outdoor reconnaissance. The robot is small and lightweight enough to be carried by any single soldier. The robot is remotely controlled by an operator who uses an operator control unit (OCU) programmed onto a commercial personal digital assistant (PDA). Each robot is completely controlled by a single OCU. The operator drives the vehicle by viewing video transmitted from a camera onboard the robot to the OCU. Such a control scheme using a virtual joystick to directly drive a robot is called direct teleoperation [1].

When operating in non-direct-line-of-sight mode, a delay on the order of one-second exists over the video data transmission channel, which happens from when the video frame is captured to when the robot responds to the operator's drive command. Direct teleoperation over a communications channel with such substantial delay is problematic for both accurate and precise maneuvering. For example, while driving the robot forward, onscreen the operator may see the edge of a cliff approaching and consequently stops the vehicle. However, what the operator sees onscreen is really what the robot captured and transmitted a fraction of a second earlier, and in real time the robot has disastrously already fallen over the edge of the cliff. Thus, any system that eliminates driving mishaps due to communications delays and also offloads responsibility from operator to robot would be desirable.

This thesis project aims to develop a real-time semi-autonomous vision-based navigation system for a small mobile robot. The eventual goal of the navigation system will be to process an operator's high level command and enable a robot to use video to navigate autonomously around obstacles enroute to the desired location. This thesis contributes an onboard payload module capable of real-time obstacle avoidance. Specifically, the system will utilize an onboard video processor to employ real-time optical flow balance calculations in order to avoid collisions with obstacles. Overall, the system will enable the operator to spend less time driving and more time planning missions and other important high-level tasks.

The system uses solely a monocular visual sensor suite for navigating. Testing of the system is targeted for indoor environments. The obstacle avoidance system was first implemented and tested in a 3-D virtual environment using an open-source graphics engine. The system was then implemented on the payload module and tested onboard the real robot in an indoor environment. In both environments, the robot was able to avoid a series of obstacles placed in its path, while some expected failure modes also manifested themselves.

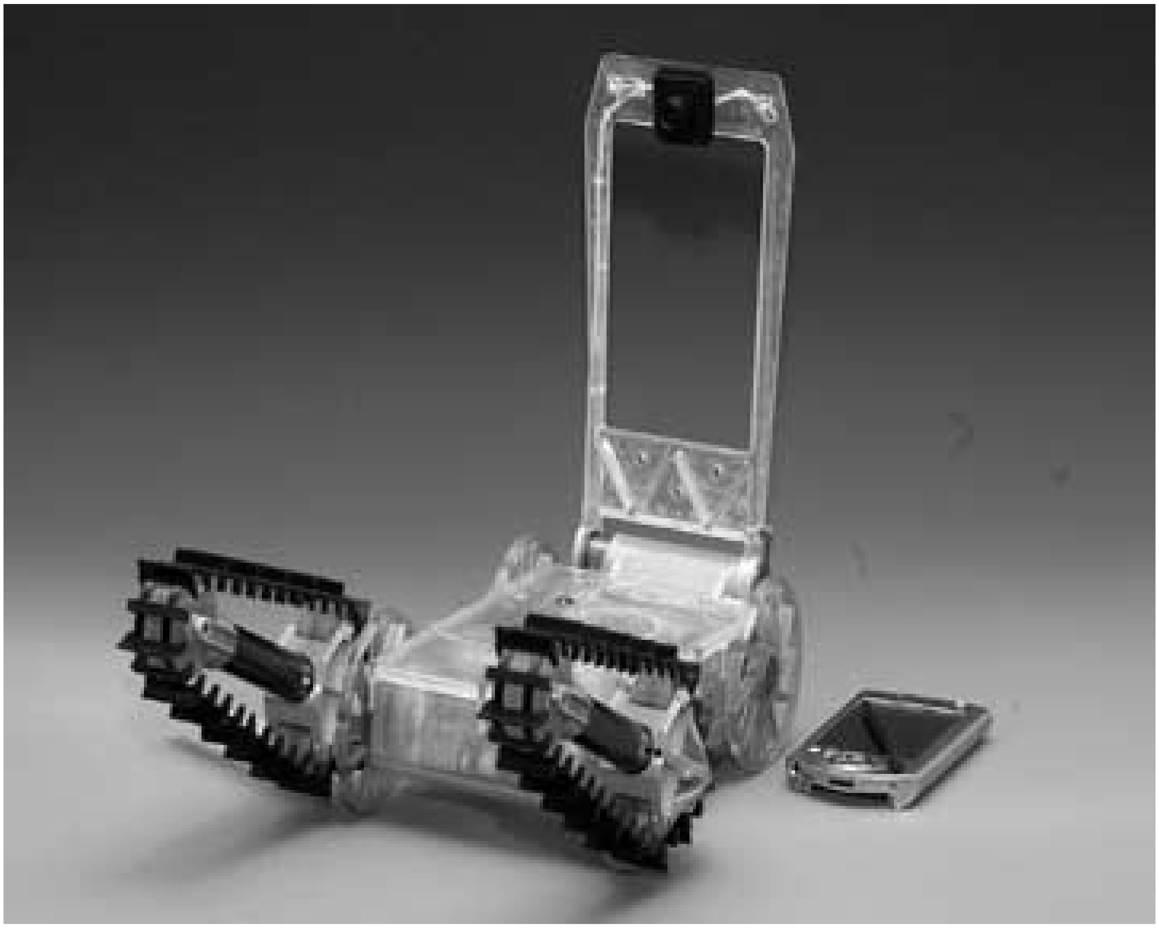

Figure 1-1: Draper Laboratory's High Mobility Tactical Microrobot. The robot is completely controlled by an operator using the IPAQ personal digital assistant (PDA) shown.

1.2 Motivation

The purpose of this thesis is to design a navigation system that provides added value and capabilities to an existing small mobile reconnaissance robot. The obstacle avoidance module is the first semi-autonomous behavior to be added as part of the system, and it is designed to run in real-time onboard the robot.

The HMTM is a recently built robot with an extensible architecture for enabling additional beneficial capabilities. The payload module in which the obstacle avoidance module resides is thus the first designed to take advantage of the payload port of the HMTM. The work done in integrating this payload module will help expedite development of future payload additions to the robot.

The general optical flow balance strategy proposed is a simple algorithm and has been shown through research literature to be a good baseline obstacle avoidance algorithm. This project allows the balance strategy to be implemented and tested first hand. The choice of this obstacle avoidance algorithm, as will be justified in more detail later, enables fast real-time obstacle avoidance with a simple implementation. Thus, the trade off between algorithm simplicity and system effectiveness is investigated through the design and implementation of the balance strategy.

1.3 Road Map

The rest of the thesis is outlined below:

- Chapter 2 describes previous work related to this thesis done in reconnaissance robotics and computer vision algorithms for mobile robot navigation. The general architecture and components of the robot are also described.

- Chapter 3 delves into the design and implementation details of the navigation system. Optical flow is defined, and optical flow computation methods are discussed. The choice of the implemented patch matching method is explained. Optimizations implemented for improving both real-world performance and robustness of the system are also discussed.

- Chapter 4 presents the results from both virtual and real world tests of the system.
- Chapter 5 presents suggestions on future work for improving and extending the system.
- Chapter 6 concludes the thesis.

Chapter 2

Background and Related Work

This thesis project aims to contribute a working module of a real-time navigation system to the growing field of reconnaissance robots. The navigation system draws from established algorithms in computer vision, specifically fast optical flow computation and template matching algorithms. The payload system relies on the established communication protocol to communicate with the host vehicle and operator control unit.

2.1 Reconnaissance Robots

The military is constantly looking for safer methods for surveying dangerous environments. The common sense goal of all methods is to remove humans as far away from the dangerous environment as possible while simultaneously acquiring surveillance data. Robots have thus been developed to acquire reconnaissance while either operating autonomously or being remotely controlled by human operators. Reconnaissance should be cheap yet effective, and thus many robotic sensor systems consist of small cheap robots that can be deployed in large numbers to form a distributed communications network of robotic sensors.

2.1.1 Unmanned Aerial Vehicles

For surveillance of distant hostile territory, satellite imagery can be employed, but unmanned aerial vehicles (UAV) can now also be deployed for aerial reconnaissance. UAV's are cheaper than satellites to develop and build, and thus more UAV's can be built and used for more reconnaissance missions. Upgrading imaging technology onboard UAV's is also easier than upgrading a satellite in orbit. Thus, though not the focus of this project, UAV's are a growing mode of reconnaissance vehicles worth mentioning.

One of the current UAV's employed by the U.S. military is the Global Hawk [2]. First flown in 1998, the Global Hawk is designed for autonomous missions, from takeoff to landing, to acquire high-resolution imagery while cruising at extremely high altitudes. The Global Hawk had logged over one thousand combat hours by June 2002. Another UAV, used by the U.S. Air Force, is the Predator [3]. Designed for cruising at medium altitudes, the Predator is operated by one pilot and two sensor operators. A color nose camera is used primarily for driving, while three other visual sensors provide visual surveillance: a TV camera, an infrared camera, and a synthetic aperture radar, which can look through clouds and haze; the three sensors are not functional simultaneously. While both the Global Hawk and Predator UAV's are large aircraft designed for distant day-long missions, other various UAV's are being designed which include micro-UAV's that are deployed in swarms for short quick surveying missions.

2.1.2 Ground Reconnaissance Robots

Ground reconnaissance robots have also been developed and put in use in the field. The Packbot is one robot used recently by the U.S. Army in the battlefield [4]. Developed by the iRobot company and funded by DARPA's Tactical Mobile Robotics (TMR) program, the Packbot has been used by ground troops in Afghanistan to scout trails ahead, transmitting video back to the operator. The Packbot is portable, able to be carried by a single soldier, and with its rugged treads for locomotion it can climb stairs and other obstacles.

In addition to the Packbot, the Throwbot has been developed by Draper Laboratory to be used for indoor reconnaissance. Also a part of DARPA's TMR program, the Throwbot is carried around in throwing mode as a handheld spherical ball. Once the robot has been thrown into the building to be surveyed, the Throwbot transforms into Spinyball mode, with a pair of spiky spokes for wheels, and an exposed camera between the wheels is used for surveillance. Both robots are non-autonomous and operated remotely through direct teleoperation.

2.1.3 Operator Control Schemes

Control of reconnaissance robots can be subdivided into three semi-distinct categories: direct teleoperation, autonomous operation, and supervisory control. Direct teleoperation has the robot under the full control of operators, and it is the simplest control scheme to design and implement. The operator drives the robot using an interface such as a joystick, and consequently the robot directly follows the operators commands. Onboard intelligence of the robot is nonexistent. Teleoperated robots allow full control to the operator for exploring environments that are too hazardous for humans. Environments such as urban battlefields with much large debris and rubble or toxic waste sites are potentially hard to navigate by fully autonomous artificially intelligent systems, and having a human operator driving the robot is beneficial for the mission.

Autonomous operation places control and decision completely within the robot. The robot is deployed by soldiers into the target environment, and the robot traverses the area according to a mission plan. For full autonomy to work, the robot needs to work as well as a human exploring the same environment: able recognize landmarks as either goals to approach or obstacles to avoid, find traversable ground, and make all decisions in real time. The difficult problem of autonomously controlled robots is continuously being researched.

Supervisory control affords the operator with high-level control while the robot possesses low-level intelligence which aids the operator. During operation, the operator executes a simple high-level command, and the robot carries it out while its programmed intelligence employs low-level tasks such as obstacle avoidance. This semi-autonomous control scheme is beneficial to the operator, because it draws an abstraction barrier between tedious low-level commands like turning left at a specified angle, and useful high-level commands like telling the robot to track a specified moving target. The robot can also be bestowed with additional programming that guides and hints the operator away from risky maneuvers and obstacles; the robot can stop itself if the operator unknowingly is driving toward hazardous terrain, or the robot can present to the operator a path through the environment that it has calculated to be free of obstacles.

2.2 Vision for Mobile Robot Navigation

Vision is the sense that enables us humans to extract the most information about the physical world, and appropriately it is the sense that we humans rely on the most. Computer vision techniques capable of extracting such information are continually being developed and refined. Vision processing is computationally intensive, but as faster and lower power processors are being developed, more real-time vision-based navigation systems for mobile robots have been implemented.

Other sensors used today for navigation include sonar, laser range finders, and inertial sensors. Sonar sensors are computationally affordable and their data is simple to read, but the reliability of their data is low. Laser range finders provide better reliability than sonar with finer directional resolution but at higher cost. Inertial navigation sensors such as accelerometers and gyroscopes provide orientation and trajectory measurements of the moving vehicle, but provide no information about the obstacles in the environment that the vehicle is traversing. All these sensors acquire less information about the physical environment than a camera can potentially, and with the continued growth of faster and cheaper computing power, that potential is now being tapped for designing real working vision based navigation systems.

For military applications, using video cameras as sensors is practical. Cameras are cheap to purchase, with even the most expensive cameras being relatively affordable. As passive sensors, cameras do not transmit or broadcast signals that may be detected by opposing forces. Overall, vision as a sensor for mobile robots is a very attractive feature.

2.2.1 Domain Specialization

Although computational power has grown, no vision system has been developed that can handle all possible environments. Most real-world systems implemented thus far have exploited specific image features of specific environments.

For example, an apparent dichotomy exists between robots designed for outdoor navigation and robots designed for indoor navigation; the specializations of these systems are such that one works well within one domain but cannot work in the other domain without considerable redesign.

Among their applications, vision systems for mobile robots are being designed for obstacle and collision avoidance, visual servoing, and path planning. Some vision systems build a map of their global environment for navigation, but such systems require much computational power and not many have been implemented efficiently onboard small vehicles using their limited onboard computers. Other systems use techniques that exploit the properties and constraints of the targeted environment; such mechanisms have been called ”lightweight vision systems.” For example, a robot exploring an indoor office environment is able to use a flat ground-plane assumption and needs only to discriminate the locally grounded free space in which the robot is safe to traverse. Lightweight vision systems are engineered specifically for each of their target environments, and although they do not in general function well for every different domain, they are meant to perform well for their defined domain.

2.2.2 Obstacle Avoidance

Simply speaking, obstacle avoidance systems can be classified into two categories: map-making systems and reactive systems. Map-making systems build a model of their global environment and plan paths accordingly around obstacles. Such systems require much computational power and data storage. On the other hand, reactive systems keep minimal state of their environment and instead rely on sensing their immediate local surroundings to figure out what areas are traversable. Such systems require less computational power and can function with robots driving at greater speeds.

For small robots, less computationally intensive and more efficient systems are desired. Horswill and Lorigo have designed robots using reactive obstacle avoidance systems for indoor environments [5][6]. Both rely on a flat ground plane assumption to visually scan for flat traversable ground terrain. Starting at the bottom of the video frame, the system reverse raster scans up the image for changes in image features, like texture or color, that signal a change from the ground to an obstacle. The algorithm is fast and effective for their environments, and it shows that vision does not have to be computationally intensive to be effective.

Another simple system uses optical flow measurements for steering around obstacles. Optical flow refers to the apparent motion of features in an image due to changing intensity patterns over time [7]. In a video stream, optical flow can be deduced from sequential image frames. From optical flow measurements, depth to objects in an image and time until collision measurements can also be deduced.

The simple system uses the fact that objects far away have optical flow fields that flow more slowly than nearby objects, although their actual motion fields are of equal magnitude. These nearby objects flow at the sides of the field of view when moving the camera past them. A balance strategy can be employed, where optical flow fields are calculated for the left and right halves of the image. When either side has substantially greater flow than the other side, it is likely and assumed to have a nearby obstacle approaching on that side, and thus it would be best to turn to the other side to avoid that obstacle. Such is the basic algorithm for the optical flow balance strategy for obstacle avoidance.

2.2.3 Visual Servoing

Visual servoing is defined as using cameras to provide closed-loop feedback control of some moving component[8]. Classically, visual servoing is a technique for controlling pose of industrial robot arms by using cameras. For mobile robots, visual servoing similarly refers to controlling a robot's orientation and location with respect to an image feature. A robot may be able to locate an object in an image, and then consequently maneuver and drive towards that object.

The earliest visual servoing systems utilized the "look-then-move" visual feedback strategy, where the vision and control systems were combined into an open-loop system. Nowadays, computing power has increased enough to enable visual servoing directly from image features. The system's job is to minimize some error function of the current image and a goal template.

For a robot seeking a goal that is dynamically defined, meaning that the goal is not known a priori, the extraction of the goal template from an image is an important concern. Template matching can be done using several methods, including SSD, correlation, and multi-resolution techniques. Occlusion and fast movement cause template matching to fail, and so the system must be able to re-acquire the goal.

2.3 Robot System Architecture Overview

The High Mobility Tactical Micro-Robot (HMTM) is designed to be a low cost robot capable of traversing a variety of terrain for the purpose of close range ground reconnaissance. The HMTM is built using commercial off the shelf (COTS) parts whenever possible, which helps by both reducing costs and expediting the design and construction processes. Figure 2-1 shows the block diagram of the data flow of the robot's constituent modules.

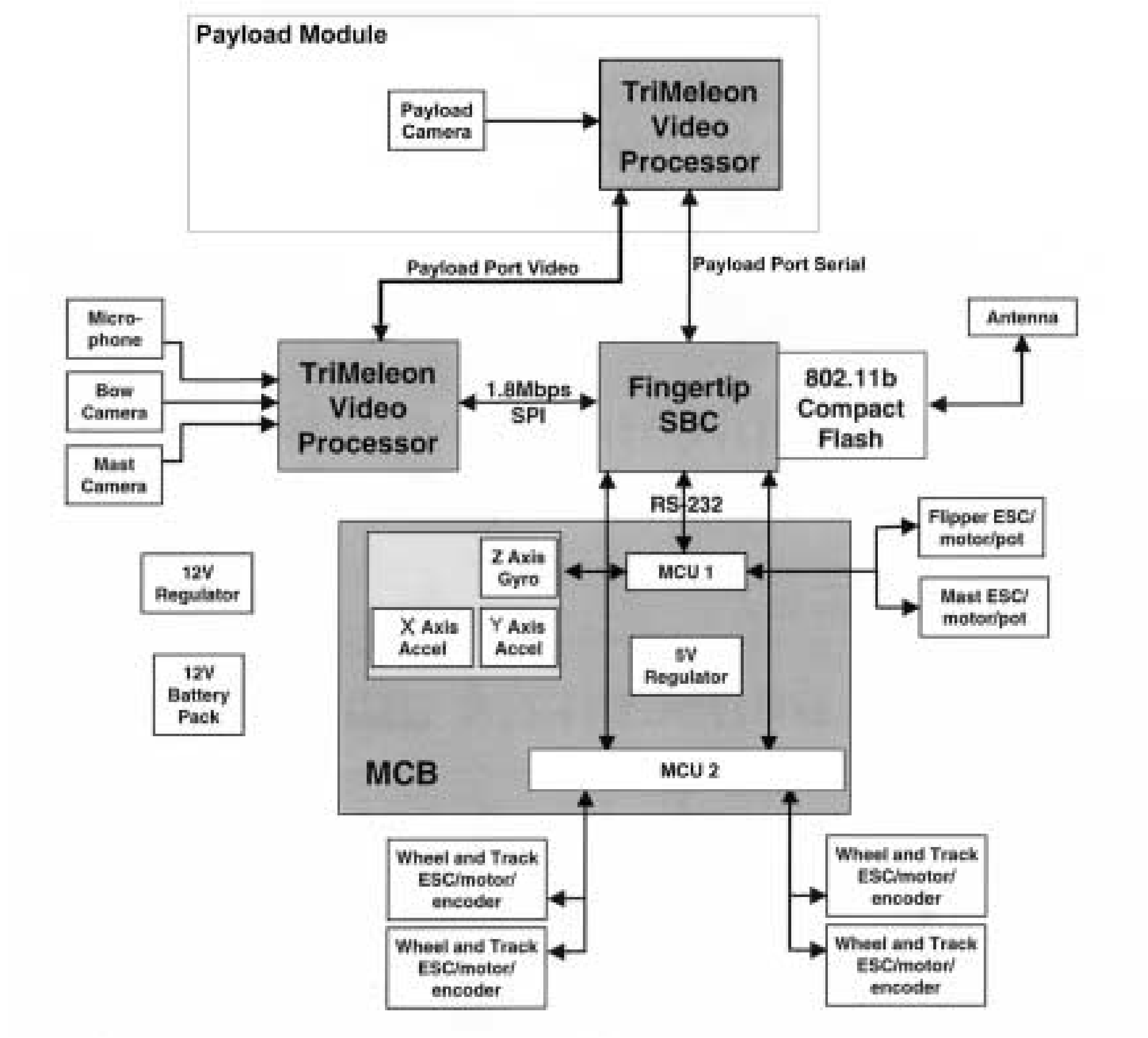
Payload Module
Payload Camera
TriMeleon Video Processor
Payload Port Video
Payload Port Serial
Micro-phone
Bow Camera
Mast Camera
TriMeleon Video Processor
1.8Mbps SPI
Fingertip SBC
802.11b Compact Flash
Antenna
RS-232
Z Axis Gyro
X Axis Accel
Y Axis Accel
MCU 1
5V Regulator
Flipper ESC/ motor/pot
Mast ESC/ motor/pot
12V Regulator
12V Battery Pack
MCB
MCU 2
Wheel and Track ESC/motor/ encoder
Wheel and Track ESC/motor/ encoder
Wheel and Track ESC/motor/ encoder
Wheel and Track ESC/motor/ encoder

Figure 2-1: Block Diagram of Robot Architecture.

2.3.1 Mechanical

The chassis of the HMTM is made of a lightweight elastomer, which provides a flexible body for better shock and impact absorption. Four elastomer wheels driven by brushless motors provide standard locomotion. The HMTM has a maximum speed of three meters per second.

A pair of articulated tracks provide additional locomotion for the HMTM. The tracks are capable of moving like flippers and have enough power to elevate the robot's body. The tracks can thus be used to orient the robot in different poses. One example pose has the tracks raising the body of the robot into an upright "prairie dog" pose. Also, the tracks can be rotated to grip on and drive over obstacles with height comparable to the height of the robot's body.

2.3.2 Electronic Hardware

The onboard electronic hardware of the HMTM consists of the following main modules: the Fingertip single board computer (SBC), the Trimeleon DSP CPU, the motor controller board (MCB), and the sensor suite. The electronics are powered by an onboard battery pack supplying twelve volts.

The Fingertip SBC is the central control module. For its CPU, the Fingertip utilizes a 206 MHz Intel StrongARM. Within the navigation system, the Fingertip's primary tasks are to forward video data from a camera to the wireless transmitter, forward operator drive commands from the wireless receiver to the MCB, and to receive serial data from the payload port. The Fingertip receives compressed video data from the Trimeleon DSP CPU and sends the data to the wireless Ethernet, 802.11B, compact flash card connected to it for transmission to the operator's PDA. For commands from the PDA to the robot, the Fingertip receives such input data from the wireless Ethernet card, and then relays the data to either the MCB for steering the wheels and tracks or selects the requested camera's video to be transmitted to the PDA.

The Trimeleon is a single board computer that utilizes the Philips Trimedia very-long-instruction-word (VLIW) CPU. The Trimeleon is designed for processing multimedia and DSP applications. In the HMTM, the Trimeleon is responsible for compressing video captured by the two onboard cameras, which are detailed in the next section. A third-party proprietary video compression algorithm is used in the Trimeleon. The compressed video is sent over a high-speed serial link to the Fingertip for transmission.

The motor controller board is primarily responsible for both velocity control for the wheels and position control for the mast and tracks. The inertial sensors are integrated with the MCB. The MCB communicates with the Fingertip via a RS-232 serial data interface.

In addition, a payload port is available for interfacing additional electronics to the robot. The payload port is located on top of the body and can support additional cameras, sensors, and actuators. The port supports serial data communication to and from the Fingertip, as well as data to and from the Trimeleon for interfacing video from an additional camera. For this project, the payload port is used to interface both a payload module consisting of an additional camera mounted on an actuator controlled mounting and an additional Trimeleon for video processing.

2.3.3 Sensor Suite

For visual sensing, the robot has two cameras onboard: one camera is installed on the main body at the bow of the robot, and the other camera is mounted on the adjustable mast and is used for surveillance from a higher elevation. Both are COTS parts and cheap to purchase. Only one camera's output is viewable by the operator at any time. The bow camera is the main driving camera. Both cameras use black-and-white CCD's, and each captures interlaced video frames in NTSC format and resolution.

The HMTM's sensor suite also includes a dual-axis accelerometer and a single axis gyroscope. The accelerometer is oriented to acquire inertial data along the X-Y axes, and the gyroscope is oriented to collect data along the Z axis, directed up from the ground plane. The inertial sensors provide dead reckoning tracking of the robot,

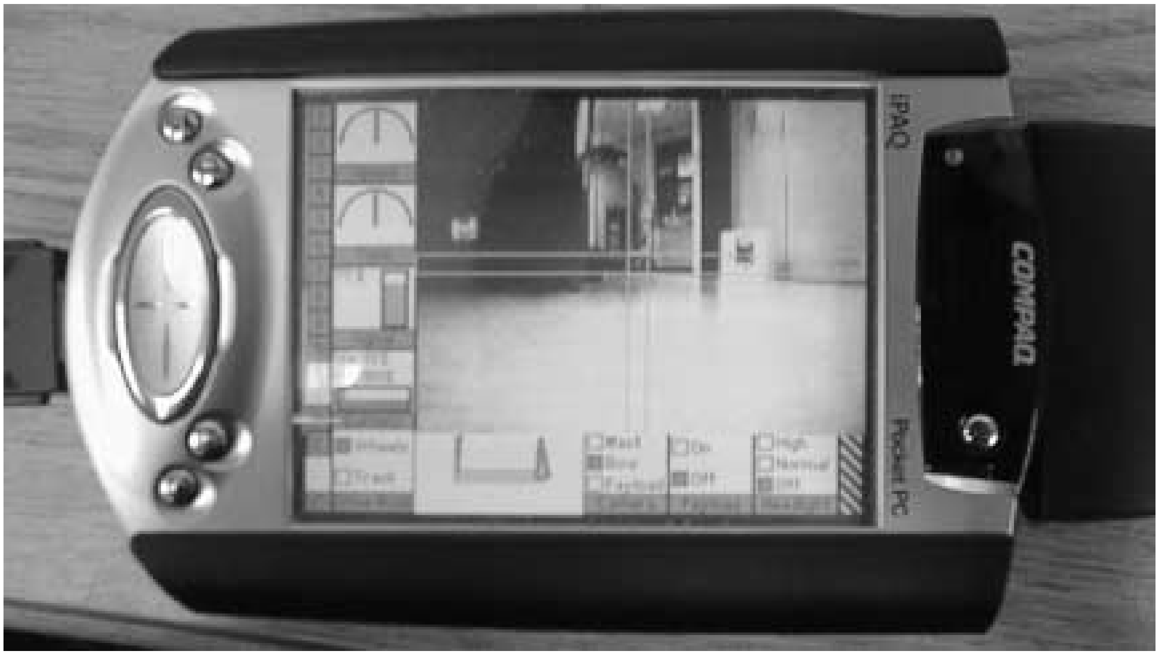

Figure 2-2: An active Operator Control Unit (OCU).

which can also be tracked from motor positioning and from speed and turn commands from the operator.

As mentioned previously, additional sensors may be added to the system by interfacing through the payload port.

2.3.4 Operator Interface

A HMTM robot is entirely controlled and driven by an operator using an operator control unit (OCU). The OCU is programmed into a Compaq IPAQ personal digital assistant (PDA). The OCU communicates with the robot through a wireless LAN Ethernet modem (802.11b); the communication protocol between OCU and robots will be detailed later in Chapter 3. Each OCU can control any one robot at a time, and each controlled robot is chosen by the OCU from the set of available robots in the field. Every available robot broadcasts its existence, and every OCU within broadcast range shows on its user display a highlighted list of available robots. Each OCU can control a robot from a maximum direct line-of-sight distance of one hundred meters.

The user interface (UI) of the OCU presents to the operator the live video captured by the onboard cameras or a payload camera interfaced through the payload port (Figure 2-2). The video is presented in 320-by-240 pixels resolution. The UI has status meters bordering the video frame; available status indicators monitor current robot velocity, steering angle, and battery life. Additional control panels are also present to adjust robot pose, mast and track positions, camera view, and drive modes.

Driving is accomplished by pointing the stylus onscreen on the live video window. The center origin of the OCU video window corresponds to zero velocity. Intuitively, the distance the stylus is dragged vertically corresponds to changing the robot's velocity; dragging up increases forward speed, and dragging down decreases forward speed. Likewise, the distance the stylus is dragged horizontally corresponds to steering the robot; dragging left makes the robot turn left, and dragging right makes the robot turn right. The stylus is also used to select the different option from the control panels on the border of the display.

Chapter 3

Navigation System

The proposed navigation system will enable semi-autonomous behaviors in the robot and consequently allow operators both to control the robots more easily and to concentrate on higher-level mission goals.

The designed navigation system relies solely on a monocular vision system to employ simple obstacle avoidance. The system is designed to be implemented on the robot's onboard video processor. However, to simplify system integration, the system is implemented in a payload module. The payload module consists of a camera and a general purpose CPU. The payload module interfaces with the host robot through a payload port which provides serial data and video data communications channels between the payload module and the host vehicle. The integrated payload and host vehicle system takes advantage of the flexible existing communication protocol for OCU and robot to communicate over the payload serial port. Figure 3-1 shows the robot with the payload module.

The following chapter discusses in detail the design and implementation of the obstacle avoidance algorithm on the payload module. Optical flow is defined, and the optical flow balance strategy that is the core of the obstacle avoidance module is described. The method of computing optical flow is justified, and optimizations for speeding up the flow computations are also detailed. Implementation problems and solutions are explained at the end of the chapter.

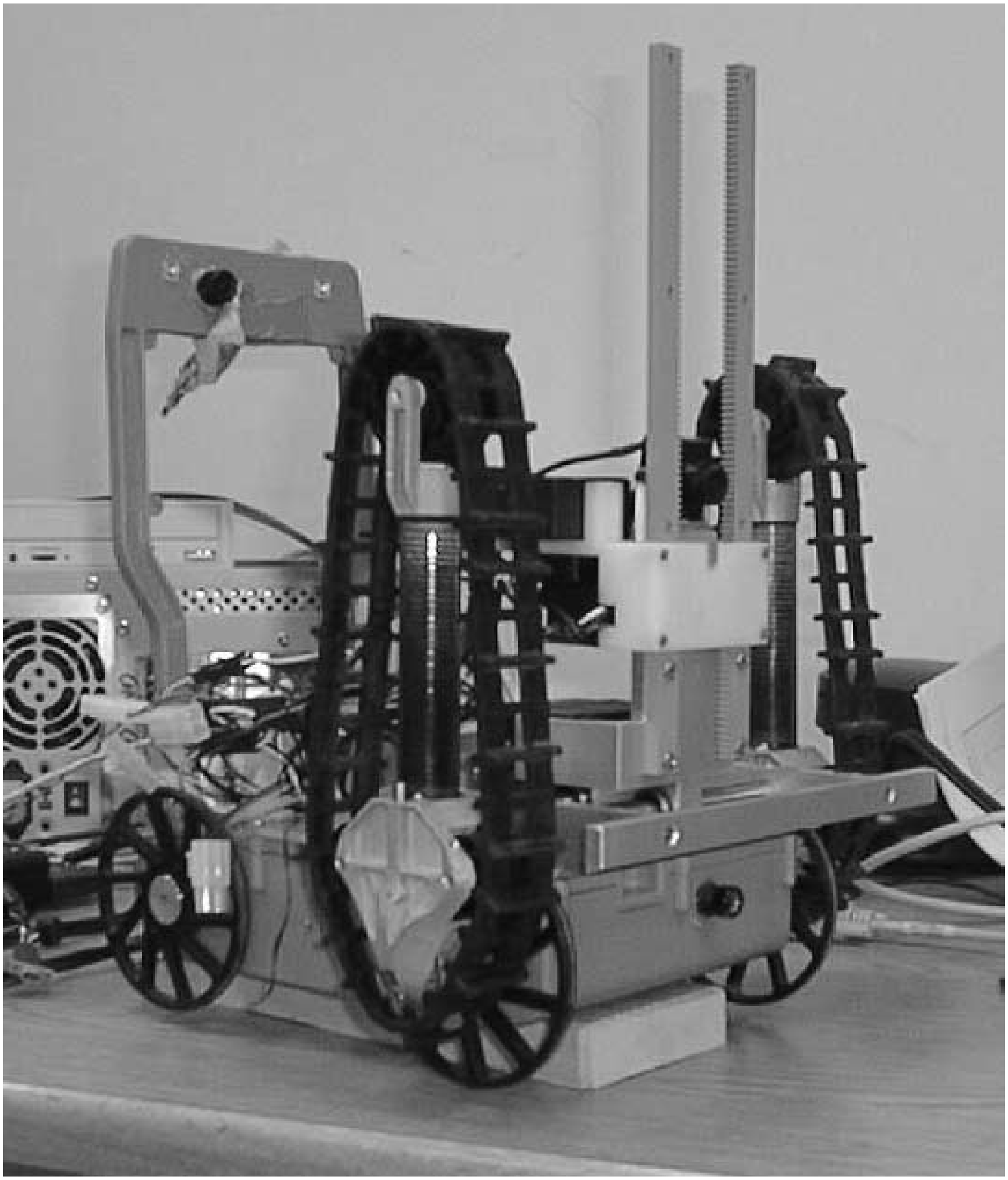

Figure 3-1: The robot with payload module.

3.1 Optical Flow Obstacle Avoidance

3.1.1 Defining Optical Flow

Optical flow is the apparent motion of the brightness pattern of an image, where the brightness pattern moves when objects in the image move [7]. The optical flow field is not always the same as the real motion field. An illustrative example is an image of a light source directed at a smooth sphere. When the sphere is rotating about its radial axis, the motion field is non-zero because the sphere is spinning, but the image brightness does not change and thus the optical flow field is zero everywhere. The sphere lacks texture that would allow for detectable non-zero optical flow and thus demonstrates the so-called ”aperture problem.”

In another case where the light source is moving around the now stationary sphere, the motion field is zero because the sphere is not moving, but the optical flow field is non-zero because of the moving light causing brightness changes. Thus optical flow fails to match the real motion field when light sources are not constant. Also, occlusions in the image, where an object near the camera overlaps an object farther away in the image, result in discontinuous optical flow. Still, except for these pathological cases, optical flow does give a satisfactory estimate of the true motion field.

Formalizing the notion of optical flow, let $E(x, y, t)$ be the irradiance, the brightness radiated from a surface point, at time t of an image point (x, y). The irradiance at time t is expected to be the same at time $t + \delta t$. Thus, $E(x + \delta x, y + \delta y, t + \delta t) = E(x, y, t)$. Assuming that brightness varies smoothly with x, y, and t, and using the Taylor series expansion, the final optical flow brightness constancy constraint equation is obtained: $\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0$.

3.1.2 Calculating Optical Flow

Many methods exist for calculating the optical flow. In general, given a pair of images of a scene taken at different times during which the robot has incrementally moved, the optical flow computed for the image pair provides an estimate of the real motion of the robot relative to its environment.

One of the first solutions to the optical flow problem was proposed by Horn and Schunk [9]. The algorithm minimizes an error function that enforces both the optical flow constraint and the smoothness of optical flow. The method can be done iteratively to refine the flow result. Unfortunately the method also can require many iterations to obtain a good result. Another method proposed by Lucas and Kanade solves the optical flow constraint through weighted least squares estimation [10]. Both these methods work with the derivatives of image intensities rather than straight intensities to obtain local estimates of flow.

A different approach to optical flow computation uses coarse-to-fine multi-resolution pyramids. The image is scaled down some N times to form a N-level pyramid. The optical flow is quickly computed first for the lowest resolution image to obtain a rough estimate. The rough result is incrementally refined by solving for the next higher resolution level of the pyramid. The method is able to obtain good flow measurements for larger search areas and thus can handle larger motion of grater magnitude.

To solve for motion globally, parametric motion models can be used. A model for motion in the image is assumed or determined beforehand, and the best matching parameters are solved to determine the optical flow. The parameters of the global transformation are determined accordingly with the assumed model, e.g. 2-D affine motion or 3-D homography. The parametric approach works well if the environment closely follows the assumed model. However such methods can currently not be done in real time.

3.1.3 Determining Optical Flow by Patch Matching

In this system, the patch matching method of computing optical flow is used. Patch matching is very general and very simple to implement. Thus patch matching can be done in real time to obtain a reasonable estimate of the optical flow.

The method finds the optical flow of discrete patches between two consecutive frames in the video input stream. The method divides up the old video frame into small patches. For each patch, the algorithm grabs an equally sized patch from the new video frame. An objective function is then used to obtain a measure of how good of a match the two patches are. The objective functions can use intensity or color values as input. Continuing on, more patches from the new image are grabbed and compared to the original patch. These patches in the new video frame are usually chosen within a search radius around the original patch's position, since it is assumed that the motion in the δt time interval has been small enough to not travel. Finally, a best matching patch as determined by the objective function is returned, and the optical flow vector is determined as the displacement between the original patch's position and the new patch's position in the image.

The objective function in this system was chosen to use just the intensity values of the input images. The choice of objective functions include taking the sum of squared differences (SSD), the cross correlation between images, and the joint entropy between images. The objective function finally chosen was the sum of absolute value (SAV) of the differences in intensity values. SAV is less computationally taxing than the other potential objective functions. Also an implementation dependent advantage of SAV is that the Trimeleon has an SAV operation built in hardware, which ensures even faster computation.

3.2 Optical Flow Balance Strategy

The optical flow of an image reflects a good estimate of the motion field of the real environment. Most of the flow computed for an image pair separated by a brief time duration is due to the relative motion of objects close to the camera or human observer. Because of the laws of perspective for mapping the three-dimensional world into our two-dimensional image plane, for a given displacement of an object, objects further away from the camera appear to move less in the image plane than objects closer to the camera. This effect can be connected with the normal everyday experience of walking down a city sidewalk. When walking forward normally, while the city skyline of distant buildings do not seem to be getting much larger in one's field of view, the people and the buildings that pass by in the local vicinity obviously appear to have much more motion.

Obstacles that need to be avoided are objects near the robot. From the viewpoint of a moving robot, the optical flow of these nearby obstacles will thus be of significant magnitude. For obstacles off to the left side of center of the field of view, the magnitude of optical flow on the left side of the image will be greater than the magnitude of optical flow in the rest of the image, on the right side. Likewise, obstacles on the right side will have greater flow than the on the left side.

The optical flow balance strategy moves the robot such that the difference of the magnitude of flow between the left and right sides of the image is minimized. By balancing the magnitude of flow on either side of the camera, the balance strategy avoids colliding into obstacles on the sides. If the magnitude of flow on the left side is significantly greater than the flow on the right, then the balance strategy dictates that an obstacle is present on the left and directs the robot to turn to the right to avoid it.

For a scenario like moving forward down a long corridor, the optical flow balance strategy promises to work well. For an obstacle centered in the camera's field of view, the balance strategy should theoretically continue moving forward until colliding with the obstacle. Changing shadows and lighting conditions will also adversely affect the desired behavior of the obstacle avoidance. Nevertheless, this simple real time system should provide a beneficial obstacle avoidance capability whose effectiveness should outweigh its drawbacks.

3.3 Optical Flow Optimizations

3.3.1 Searching for the Best Patch Match

For each patch in the original image, searching for the best matching patch in the new image is a time consuming process. For each patch of size N-by-N pixels, the patch matching search over a square region with sides equal to M pixels needs M^2 patch comparisons, with each patch match needing N^2 absolute value computations. For this system with neither a fast processor nor an operating system with multithreaded capabilities, patch matching is the primary performance bottleneck.

To improve performance, the patch size and search region size can be reduced. Too small of patches however do not have enough information to produce good results and instead produce more false positive matches. Decreasing the search region size also can improve speed of performance by reducing the number of potential patches to compare. A smaller search area however limits the maximum magnitude of motion that the algorithm can detect. Since faster motion results in larger magnitude optical flow vectors, the patch matching algorithm cannot find optical flow vectors with magnitudes greater than the search radius.

A method that reduces the number of patch comparisons, preserves an adequate search radius and patch size, and still results in producing a good patch match is a logarithmic search [11]. Whereas the exhaustive patch matching search described thus far compares the original patch to every potential patch in the search area, logarithmic search works by testing only nine potential patches at a time. The nine patches are chosen from eight points on the border of the square defined by the search radius and also the center of the search area. For the intermediate best match determined from those nine points, a new search area is defined to be centered on the best matching point. The search radius is then reduced in half, and a new set of nine patch candidates are chosen, with the border points chosen as being the new halved radius distance away from the new center point. The search ends when the radius is reduced to one, and the best matching patch found at the end of that iteration is the final result. The computational savings for patch matching with an $M = 9$ search radius is the substantial difference of the total $M^2 = 81$ patch comparisons for the exhaustive search and $9\lfloor log_2 M \rfloor = 27$ patch comparisons for logarithmic search.

The log search does not guarantee that the globally optimal patch match is obtained as the result. Nevertheless, the best result can be assumed with good confidence to be achieved. An additional confidence check of the final result can be done by a final pixel-by-pixel comparison of the original patch and the best matching patch. Each pixel is checked to be within a threshold difference in intensity from the corresponding original patch pixel and if so is labeled a good pixel. Then only if the number of good pixels is above a set threshold, then the best matching patch is determined to be a valid match. Otherwise, the patch is deemed invalid, and the optical flow vector is zeroed for that patch position.

3.3.2 Domain Specific Optimizations

Although this system is targeted for any environment in general, both indoor and outdoor, some constraints and simplifications about the robot's behavior and its usual environments can be exploited to improve optical flow computation time and overall performance.

The robot for the most part will be traveling on a level ground plane. Indoor reconnaissance will most definitely be on flat ground, while outdoor reconnaissance is less likely but most probably to be on flat ground. The obstacles that the robot will encounter are most likely standing on the ground. The magnitude of the optical flow vectors determined thus far are two-dimensional for the image plane. The robot however will be moving on the level ground and avoiding obstacles to its left and right, along the horizontal x axis of the image. Any flow along the y component is less important to obstacle avoidance in this robot's case. Excessive flow along the vertical y component can even cause the robot to react negatively to the flow imbalance, when the vertical flow should not be considered a nearby obstacle to avoid. A case where the y component of flow dominates the x component is when driving towards a tall distant object, which grows more vertically than horizontally in the camera's field of view, and which for correct operation ought not to influence the avoidance of nearby obstacles. Thus, instead of taking the magnitude of the actual optical flow vector, only the magnitude of the x component of the flow vector is used in the balance strategy.

In addition, because the vertical component of optical flow is not as important to the robot, the image size can be scaled down vertically to create a smaller image. Therefore less optical flow computations are needed and the speed of the algorithm is improved. The loss of image information along the y axis is a good tradeoff for faster processing, because of the simplification that flow along the horizontal x axis matters the most and will still result in a good estimate of the actual optical flow of the image pair.

3.3.3 Other Potential Improvements

Many other potential improvements were considered to both improve the correctness and the speed of the algorithm. All of these mentioned though cannot be done in real-time on the given hardware and thus were not integrated into the design of the system.

The current optical flow algorithm accounts for only integer pixel motion. Fractional pixel motion however is a more accurate model. The method of bilinear interpolation is often used to estimate fractional pixel flow.

Edge detection, or some other image feature filtering operation, can be done prior to flow computation. Edges and corners are strong image features that the algorithm relies on for accurate optical flow. By filtering out the strong edges and zeroing the rest of the image, the random noise problem present in the details of the original images can be solved. Edge detection needs to be robust and accurate, to ensure that the edges representing a single object are fully connected in the filtered image.

3.4 Robot and OCU Communication Protocol

The communication protocol between robots and the OCU's rely on packets of data sent serially over wireless LAN. The packets transmitted by an OCU to a robot are used to initially gain control of a robot and then to manipulate the robot through operator commands. The packets transmitted by a robot to an OCU are used to broadcast to all OCU's that the robot is available to be controlled, to respond to an OCU's request to take control of it, and to report its current state to the controlling OCU.

Packets are defined to be of two types: heartbeats and NOPs. As the name implies, heartbeat packets are sent periodically by a live OCU or robot, specifically

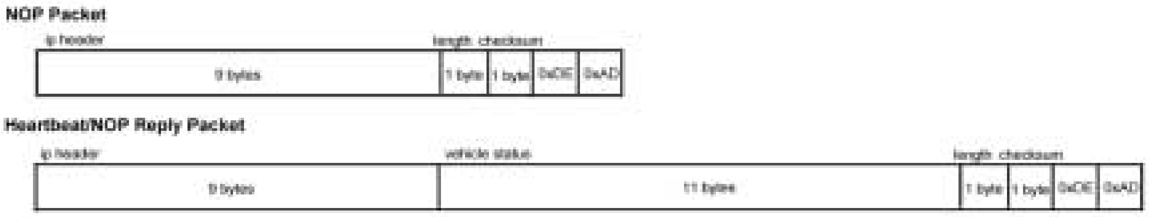
NOP Packet
ip header
length
checksum
9 bytes
1 byte
1 byte
0xDE
0xAD
Heartbeat/NOP Reply Packet
ip header
vehicle status
length
checksum
9 bytes
11 bytes
1 byte
1 byte
0xDE
0xAD

Figure 3-2: Packets used in the OCU/robot communication protocol.

both by a controlling OCU and an available robot. Heartbeat packets contain both an IP header portion and a vehicle state portion; the packets contents are detailed in Figure 3-2. The IP header provides data to ensure that the packet reaches the intended destination. Every robot has a unique robot identifier and every OCU has a unique OCU identifier. The vehicle state either gives data of a robot's current state, if the heartbeat is transmitted from a robot, or is a command to a robot to reach this desired vehicle state, if the heartbeat is transmitted from an OCU controlling a robot.

NOP packets are transmitted by an OCU to gain control of a robot. The NOP's sent by an OCU consist of only an IP header. The NOP is also sent by a robot to reply to an OCU; this NOP has both an IP header and a vehicle state portion.

Every heartbeat and NOP packet has a trailer consisting of the length of the packet and a checksum. The checksum is just the XOR of all data bytes, excluding the length field.

The complete process of establishing communication between OCU and robot is as follows:

- When a robot is turned on, it begins transmitting a heartbeat packet to a unique broadcast destination. All active OCU's in range parse the broadcast packet and determine that the robot who sent the packet to be active and available to be controlled.

- When an operator selects a robot to control, the OCU begins transmitting NOP's with the target vehicle as the destination. The OCU continues transmitting the control NOP until the robot returns an acknowledgment NOP. The robot stops sending broadcast packets after it has been under control.

- After the OCU gains control of the robot, the OCU periodically sends heartbeats with the latest control variables to manipulate the robot. The robot responds to each OCU heartbeat with a heartbeat of its own transmitted to the controlling OCU.
- When the OCU wants to stop controlling the robot, it sends a heartbeat which sets the state of the robot to standby, which the robot recognizes as the end of the OCU controlling it.

3.4.1 Payload Module Control

To enable the payload module to control a vehicle, the packet communication protocol was extended. When the payload port of a host vehicle is activated, the host automatically echoes all packets it receives from the UDP port for receiving wireless LAN data over the serial data port of the payload port, and it also echoes all packets it receives from the payload serial port over the UDP port. The payload module is now privy of relevant communication between OCU and its host and is able to send packets wirelessly via the host vehicle to OCU's.

Taking advantage of the existing protocol, the payload module becomes both a simulated vehicle and a simulated OCU. As a simulated vehicle, the payload needs to obtain a vehicle ID and to broadcast itself to OCU's to inform them of its availability. The vehicle ID is obtained by finding the host vehicle's ID from any packet that the payload receives and then adding one to the host ID. This does not ensure that the payload vehicle ID is unique, so care must be taken to make sure that no other vehicle has such an ID. Thereafter, whenever the payload receives a broadcast packet from the host, it sends its own broadcast packet via the host and thus becomes an available vehicle to be controlled.

As a simulated OCU, the payload needs to gain control of its host vehicle whenever an OCU wishes to gain control of the simulated vehicle. When the payload receives a

NOP request for control from an OCU, the simulated OCU is activated. Accordingly, the simulated OCU, which has its specifically assigned unique OCU ID, sends its own NOP request for control to the host vehicle. The simulated OCU continues sending NOP requests until the host replies. Now that the host is under the control of the payload, the simulated vehicle can acknowledge to the OCU that it is ready to be controlled, and it replies with a NOP to the OCU. The payload is now a middleman between OCU and host vehicle communications: it can echo OCU control packets as need be when its obstacle avoidance module finds there to be no obstacles in the way, and it can change the contents of the control packets before forwarding them to the host when its obstacle avoidance module detects an obstacle nearby and issues an avoidance maneuver.

3.5 Implementation Details and Optimizations

3.5.1 Imbalance Thresholding and Translating Flow Imbalance to Drive Commands

The balance strategy outputs a ternary result to the robot: either the flow is balanced, or the flow is imbalanced with greater flow on the left, or the flow is imbalanced with greater flow on the right. To achieve the classification boundaries for these three cases, a flow difference threshold is set. If the absolute value of the difference of flow magnitudes is less than the threshold, the flow is considered balanced. If the absolute value of the difference of flow magnitudes is greater than the threshold, then the flow is classified as imbalanced.

The output of the obstacle avoidance module implementing the optical flow balance strategy affects the robot only when the flow is imbalanced. To translate the calculated flow to a potential drive correction command, the following function determines the output of the flow module:

$flow = \frac{(left\_total\_flow - right\_total\_flow)}{(left\_total\_flow + right\_total\_flow)}$

The result of $flow$ is a value in the interval $[-1, 1]$. For $flow < 0$, the flow is imbalanced with significantly greater flow on the right side, while for $flow > 0$, the flow is imbalanced with significantly greater flow on the left side. Zero is assigned to $flow$ when the flow is balanced. The result is thus proportional to the magnitude of the flow difference; for a slight imbalance, $flow$ is near zero, while for a large imbalance, $flow$ has a larger magnitude toward ± 1.

To correct for flow imbalance, two drive commands, one for each side of wheels of the robot, are needed. The robot is driven with two independent motors for the left pair and right pair of wheels. When the flow is imbalanced, the current OCU drive commands need to be corrected. The new drive command takes on the following value:

$new\_drive\_left = old\_drive\_left + K * flow$

$new\_drive\_right = old\_drive\_right - K * flow$

K is a turn multiplier that is implementation specific and needed to produce a correction in the valid range for a drive command.

3.5.2 Image Noise Reduction

Real world image processing must be robust to handle and reduce noise that corrupts image quality. Whether caused by physical connections from the video input from the camera or interference from other electromechanical components in the robot, noise in the images is sure to reduce the accuracy and effectiveness of the image processing algorithm which assumes the best of conditions.

Figure 3-3 shows two images, A and B, which were taken by a stationary camera at a δt time apart. The scene is of white pieces of paper taped to a stationary wall with a fluorescent light shining from the left. Image B shows the x components of the optical flow vectors calculated by patch matching; many of the flow vectors are non-zero and thus incorrect because both the scene and camera are stationary. So even though the two images appear the same from a casual inspection, the actual difference of the two images, shown in C, looks to be random shotgun noise. The noise can be attributed to both noise in the video system, and the real world light source and the real world non-Lambertian surfaces that do not reflect a constant amount of light at

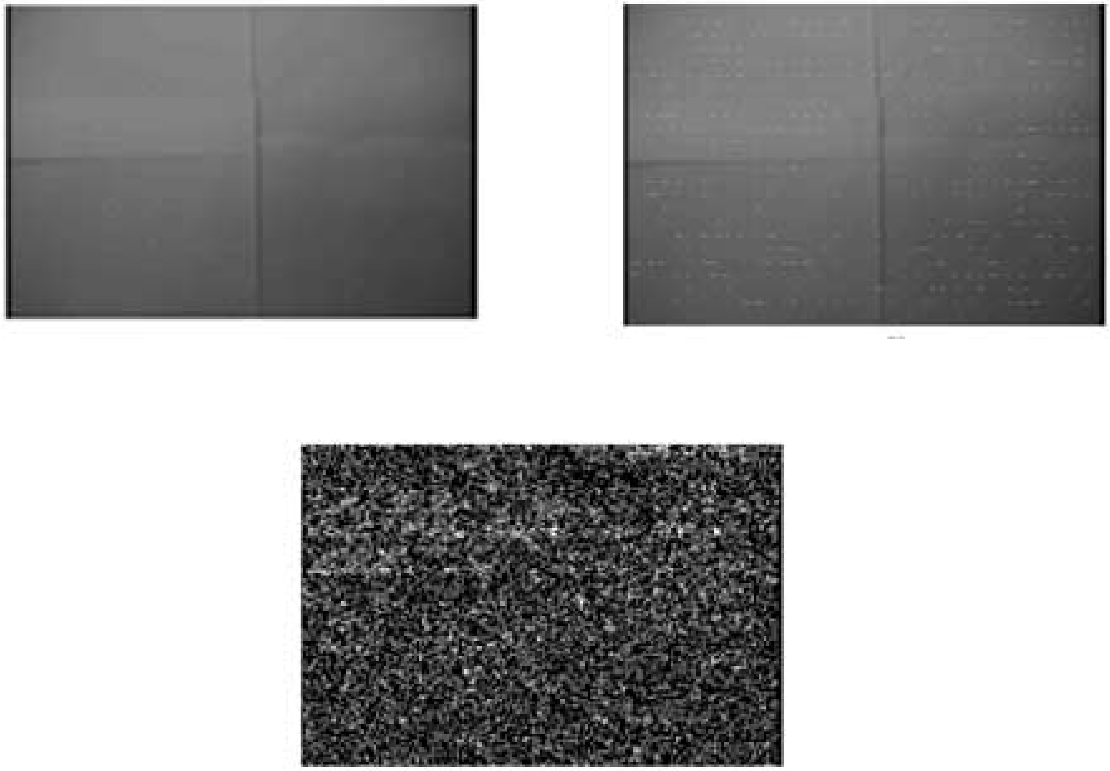

Figure 3-3: Image pair of a stationary scene and their noisy difference image. The top images show the same stationary scene acquired by a non-moving camera. The bottom image shows the difference of top image pair. The noisy non-zero difference image C presents a problem for accurate optical flow computation between the image pair.

any instance of time. Whatever the case, calculating the optical flow between a noisy image pair of a completely stationary scene will produce incorrect results.

To reduce the negative effect of image noise, both a smoothing filter with a Gaussian kernel and a median filter were first tried. Unfortunately both were computationally intensive and both did not result in adequate noise removal.

The solution implemented relies on balancing the difference of the image pair. The noise is assumed to be random and uniformly distributed. Thus for a stationary image pair, like shown in the example, their difference image is assumed to be of random intensities pulled from a small range of near-black intensities. Like the optical flow balance strategy, this noise solving method computes the SAV of the intensities of both the left and right sides. If the image is stationary and the assumption of uniformly distributed noise is valid, then the difference between the left and right sides of the difference image will be small. If the image pair is of a scene with significant motion, then the difference between the left and right sides of the difference image will be noticeably larger. Therefore a threshold can be set for which differences greater than that threshold are determined to be caused by image motion. This difference image balance strategy is incorporated with the optical flow balance strategy, such that both the difference in image intensities and the difference in optical flow magnitudes of the left and right sides of the image must exceed each method's respective threshold for the flow to be deemed as imbalanced and in need of correction by the robot's drive system. Also, this difference image balance strategy adds only a constant factor to the total running time of the system, because the difference image is already computed as part of the patch matching part of the algorithm.

Figure 3-4 shows an image pair with significant motion on one side of each image. The absolute difference image has significantly more magnitude for the right side, and the magnitude of optical flow vectors is also greater on the right side. Thus, this image pair will alert the obstacle avoidance module to avoid the assumed obstacle on the right side.

3.5.3 Time Averaging Flow

Successive optical flow balance estimates may be discontinuous; that is, fast moving objects in the field of view may cause the balance strategy to cause the robot to vacillate between turning left and right. Spurious flow balance results may also result in the robot to twitch left and right as it moves forward.

To dampen such erratic turning behaviors, the optical flow balance result is actually an average in time of the past N results, where N is a changeable input parameter. Too small of N and the effect of averaging is non-existent, and too large of N and both the latency of a flow imbalance actually triggering a drive correction is great and any brief but significant durations of flow imbalance may be averaged out to under the imbalance threshold and consequently incorrectly classified as balanced.

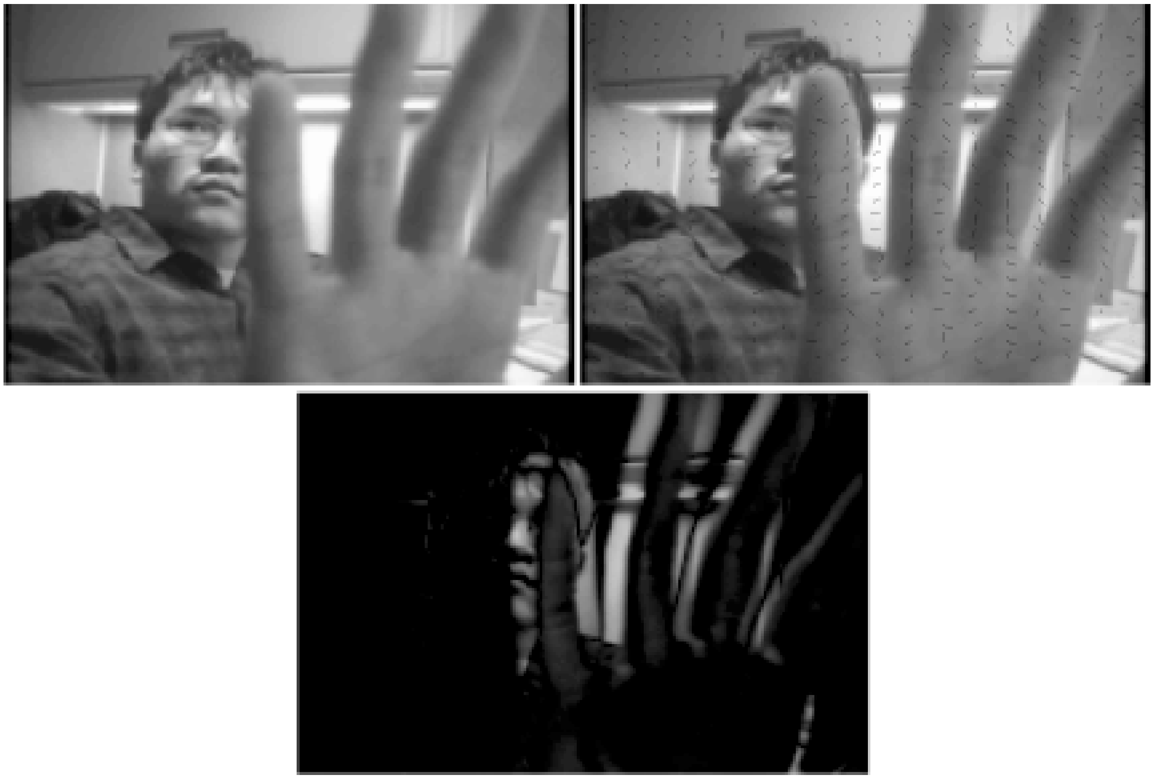

Figure 3-4: Image pair of a scene with motion and their absolute difference image. The top two images show the subject moving his hand between the two frames. The bottom image shows the absolute difference of the image pair, with large magnitudes on the right side where the hand moved. In this case, the image difference imbalance is great enough to mask image noise.

3.5.4 Program Timing Constraints

The obstacle avoidance module was programmed in C and run on the payload module's Trimeleon. The program on the Trimeleon is run without an operating system. Although a real-time operating system for the Trimeleon became available late in this project, unfamiliarity with the software and the substantial amount of time needed to integrate the OS with the system were strong factors in deciding against using the OS presently. Thus the program is run sequentially and single threaded.

The main loop of the program manages the pulse width modulation (PWM) for controlling the payload servos, the serial data communication between the payload and the host vehicle, and the optical flow calculation for sequential video frames. The period of the main loop is dictated by the tightest timing constraint of these three processes. The servo motors used have a PWM period of 20 ms, and the control scheme implemented requires updating servo control parameters in half that time. Although the minimum 10 ms for servo control is the most limiting constraint in time, a timer interrupt service routine controlling the PWM is able to preempt other processes, and thus its time constraint is always satisfied.

The serial buffers though can hold at most 64 bytes. Since additional bytes transmitted to a full receiving buffer are dropped and lost, buffer overflow must be prevented for correct operation. The serial UART communication operates at 115200 bps. With 10 bits sent over the serial data channel per byte, and assuming the worst case of a continuously transmitted serial data stream, the 64-byte buffer is filled in 4 ms. Since each cycle processes one received serial data packet, the cycle time is limited to at most 4 ms.

The total time for optical flow computation between two successive frames is substantially greater than the maximum cycle time. The patch-matching algorithm running on the Trimeleon processes 10 to 15 frame pairs per second, when no other processes are running on the Trimeleon. Without an operating system and multi-threaded capabilities on the Trimeleon, the optical flow computation needs to be divided into sub-problems that are each able to be solved in one cycle. Accordingly, the flow computation is divided into sub-problems of matching a single patch between video frames. There now exists a constraint on the patch size that can be compared and processed within the serial buffer overflow time constraint; this can be unconstrained by subdividing each patch into smaller subproblems of calculating flow for a fraction of the patch.

3.6 Supervisory Control of Robot

With the obstacle avoidance module implemented on the payload module, and the communication protocol defined for OCU to simulated vehicle and simulated OCU to host vehicle, the system is ready to help the operator.

Figure 3-5 summarizes the control flow between the operator and his or her OCU and the obstacle avoidance module. The control scheme of the robot by the operator is that the operator directs the robot to drive directly forward. Given non-zero forward drive commands, the obstacle avoidance module is activated and makes drive corrections as needed when the optical flow becomes imbalanced. The obstacle avoidance module only reacts when forward drive commands are sent by an OCU to the simulated payload vehicle. Thus, the operator is in more of a supervisory control mode than full direct teleoperation of the robot. However, he or she still determines when the robot moves or stays stationary, and in that regard, a lesser degree of semi-autonomy is achieved: the robot is just a helper to the operator, rather than the operator being a helper to a more capable and more autonomous robot. Nevertheless, that the obstacle avoidance system can prevent the operator from inadvertently colliding into obstacles is a significant benefit, and the system is a good first step towards evolving more complex autonomous behaviors in the robot.

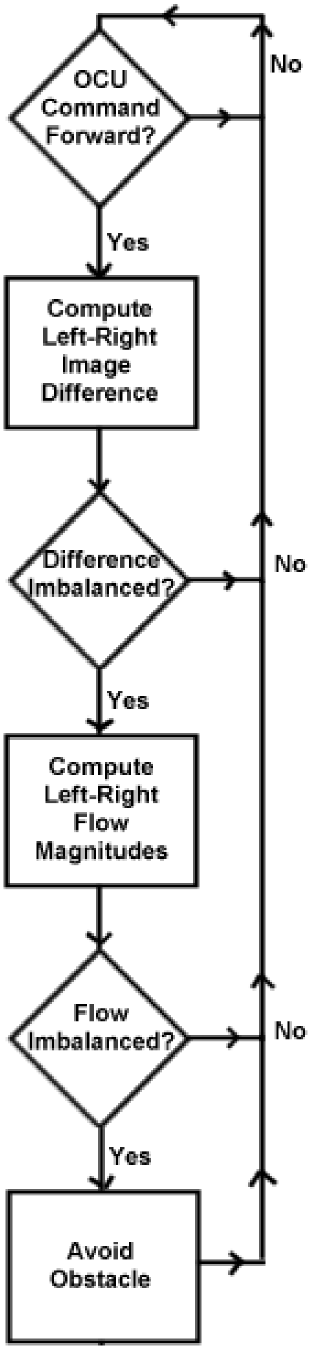
OCU Command Forward?
No
Yes
Compute Left-Right Image Difference
Difference Imbalanced?
No
Yes
Compute Left-Right Flow Magnitudes
Flow Imbalanced?
No
Yes
Avoid Obstacle

Figure 3-5: Flowchart for deciding between operator and obstacle avoidance control.

Chapter 4

Testing and Results

The obstacle avoidance system was tested in both virtual and real world environments. Different parameters of the system such as patch size and flow thresholds were input and tested.

The obstacle avoidance module was first implemented and tested within a virtual environment created by a 3-D graphics engine. The system was then implemented on the payload module and tested onboard the host vehicle. The system was tested in an indoor lab environment. The robot was directed to travel around everyday obstacles, including tables and chairs and people.

Overall, the system performs as well as can be expected from the simple optical flow balance strategy. The obstacle avoidance capability is definitely a beneficial augmentation of the system. Further higher-level improvements can next be built on top this system to achieve the eventual goal of a general semi-autonomous navigation system.

4.1 Simulated Environment

4.1.1 Setup

The obstacle avoidance system was initially implemented and tested within a virtual environment. Using an open-source 3-D graphics engine, a virtual world was created for a virtual robot to navigate.

The graphics engine Crystal Space was chosen for simulating the virtual environment [12]. The engine is free open-source software and was thus configurable to the needs of this project. A software renderer was used within the graphics engine; although a more powerful hardware graphics system was available on the personal computer that the virtual environment was run on, the parameters and settings were not able to be configured to run properly, and thus the hardware-supported video mode was not used. Consequently, the lighting, reflectance models, and texture details of the virtual environment were not as accurate as they could have been.

For all tests, the virtual robot was let loose to freely roam a series of maps of virtual environments. Collision detection was not implemented in the virtual world, and thus the virtual robot was able to unrealistically travel through solid objects and walls. Nevertheless, testing in the virtual environment did allow the balance strategy to be evaluated.

The parameters of the optical flow computation, most importantly the patch size and search region size, were varied to find the best set for the best performance. Due to computing resource constraints, the size of the patches was limited to below a small maximum size.

The maps that the robot explored included several pre-made maps included with Crystal Space and two custom-made maps. The simplest map consisted of a regular grid of interconnected rooms without obstacles. The pre-made maps included a large indoor complex with several interconnected rooms with variable lighting conditions and long corridors. An urban outdoor environment was also available. The custom maps were both made up of two rooms connected by a long corridor.

4.1.2 Results

The balance strategy was successful in controlling the robot to travel down corridors, avoid a series of regularly and irregularly shaped obstacles, and follow along walls. In general, the system was adept at traveling down long corridors with lightly textured walls. The virtual robot was able to follow the textured walls around a room and was

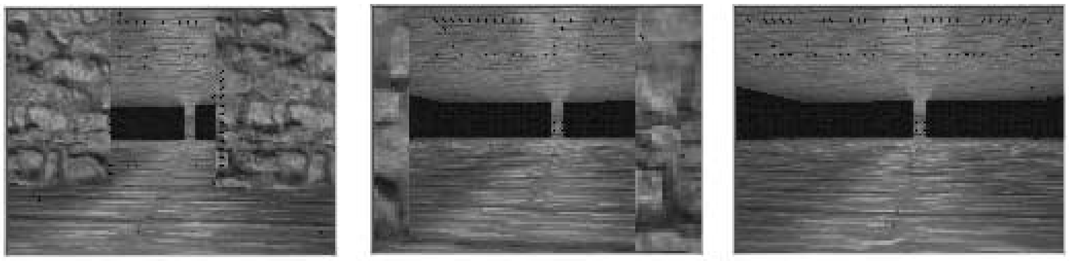

Figure 4-1: Successive frames of the virtual robot going through an open doorway.

able to avoid corners fairly consistently. The obstacles in the room were scattered at adequate spacing such that the robot could react to avoid the nearest obstacle and not run straight into another obstacle.

In the simplest environment, the robot was let loose to roam empty rooms connected in a grid array pattern with open doorways. The texture of the walls of the rooms were changed between a high contrast mosaic texture and a lower contrast stone brick texture. The floor and ceiling were of a wooden plank texture. The performance of the robot in these open rooms showed ability to follow the walls around a room. The robot was able to traverse through an open doorway to an adjacent room (Figure 4-1). When traveling in the middle of each large room, the robot would often see the corner of the room in front of it, and that sharp edge exhibited enough optical flow for the robot to turn away from that potential "obstacle." The regular pattern of floorboards also influenced the robot's direction, as it on occasion aligned itself to follow along the length of the edges of the floorboards. While driving toward a mosaic wall, the robot would center itself on the large fan-like mosaic pattern, like it was a doorway; thus the robot collided into the wall, as shown in figure 4-2.

Figure 4-3 shows frames of the virtual camera roaming a different environment, one consisting of two rooms connected by a long hallway. One room where the robot starts from is empty, and the other room is filled with obstacles of different shapes and textures. The figure shows the different shapes of obstacles placed in the robot's way. The walls and obstacles were mapped with high-contrast textures to allow for

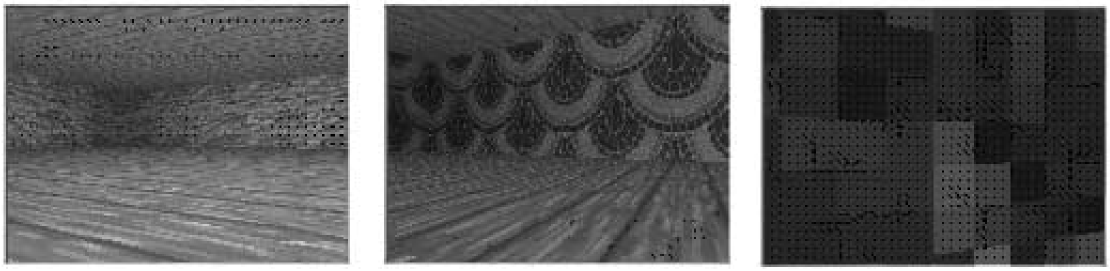

Figure 4-2: The virtual robot roaming through open rooms. The left frame shows the robot approaching the corner of the room with stone brick walls. The middle frame shows the mosaic pattern walls. The right picture shows the robot's field of view after colliding into the mosaic wall. Notice that all the optical flow vectors and their respective patch centers are visible.

less ambiguous optical flow. The frames also are overlaid with the optical flow vectors calculated for that frame. Unfortunately colored in black, each flow vector emanates from the center of the patch used in the patch matching optical flow method. The frames thus show the density of the flow vectors used to achieve the results in the virtual world.

Again, the robot successfully navigated its way down the hallway. The robot did not approach any obstacle head-on initially, and thus was successful in avoiding them.

As for some shortcomings, while driving down the corridor, the virtual robot did weave left and right a bit erratically as it maintained its flow balance. The lack of collision detection meant that on occasion, when the robot followed a wall into a corner, the robot failed to turn fast enough and ran through the wall. The highly textured environment is not representative of the real world and neither are the aliased and jagged edges drawn by the graphics engine. Dark shadows with sharp edges were incorrectly determined as obstacles and thus were avoided by the robot although open space in front of the shadow was available. Also, after wandering the room for a longer period of time, the robot came to the situation where its flow was balanced, but an obstacle was situated in the center of its field of view; the obstacle stayed in the center of the FOV until the robot collided.

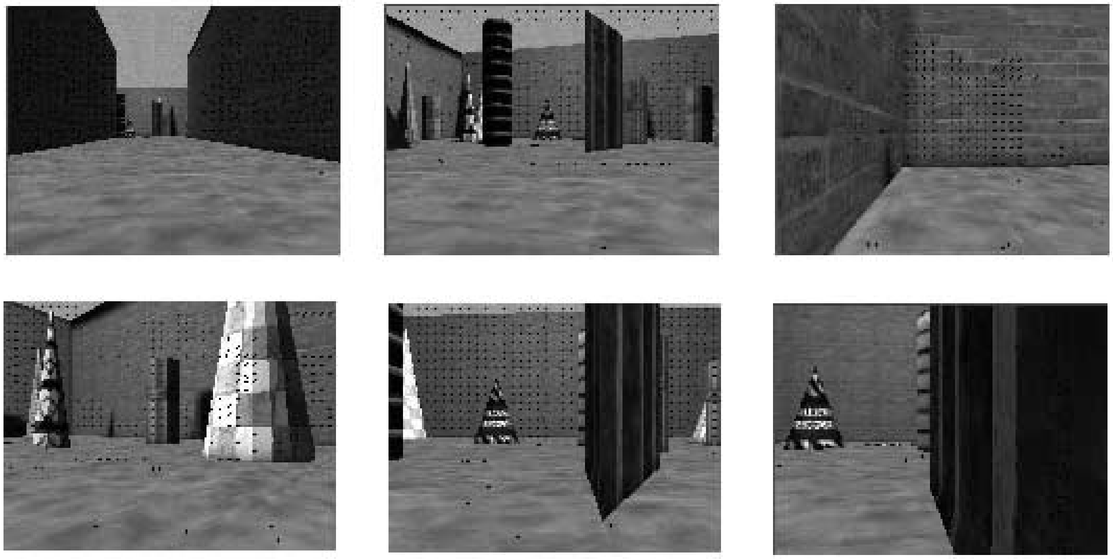

Figure 4-3: Frame captures of the virtual robot roaming in a virtual environment. The map is laid out as two rooms joined by a long corridor. The top-left frame shows the virtual robot traveling down the hallway. The top-right frame shows the virtual robot following along the brick-textured wall. The rest of the frames show the virtual robot encountering various regularly shaped obstacles. All the frames are overlaid with all non-zero optical flow vectors, which are colored in black. Each flow vector emanates from the center of a patch used to compute optical flow.

4.2 Real World Testing

4.2.1 Setup

The payload module and host robot were tested indoors in Draper Lab. A simple slalom course of obstacles was setup for the robot to negotiate; the obstacles include tall chairs, buckets, and stationary people. For the most part though, the lab environment was not setup in a contrived fashion for testing. Rather, the robot was made to navigate around the obstacles as they would lie around the lab normally. Such makes the testing more real world applicable. Also, the robot was run through both open areas and more cluttered areas, to test the balance strategy's performance with obstacles at varying distances from the camera.

The payload camera is mounted on the servo-controlled platform. Because the servos are not needed for this task, the servos were left turned off, and the camera was positioned at the setting closest to the ground, to achieve a viewpoint as near the body of the robot as possible. The camera is also mounted on a rotating head that can be tilted vertically. Thus, to achieve a better view of the ground immediately in front of the robot, the camera is tilted down a few degrees below level.

4.2.2 Results

The robot was able to avoid single obstacles on either side of its field of view successfully. The obstacles included orange buckets, swiveling office chairs, and metal table legs. Figure 4-4 shows the robot navigating between the small space between a bucket and a table leg. Figure 4-5 shows the robot avoiding a wheeled cart in the middle of the lab. Figure 4-6 shows the robot avoiding both an office chair and desk drawers.

The expected failure modes with textureless obstacles and obstacles centered in the field of view did occur. When negotiating an open passage between a table leg and a plastic container, the robot turned away from the table leg and ran into the plastic container. The plastic is a solid off-white color and smooth texture, thus giving little

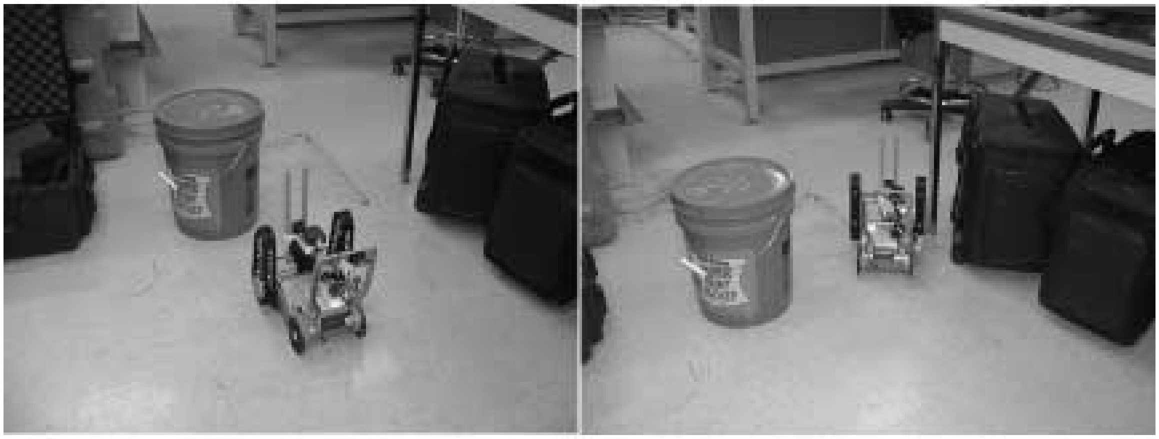

Figure 4-4: The robot driving between a bucket and a table. The robot approaches the bucket (left figure), turns right to avoid the bucket, and drives between the bucket and the table leg (right figure).

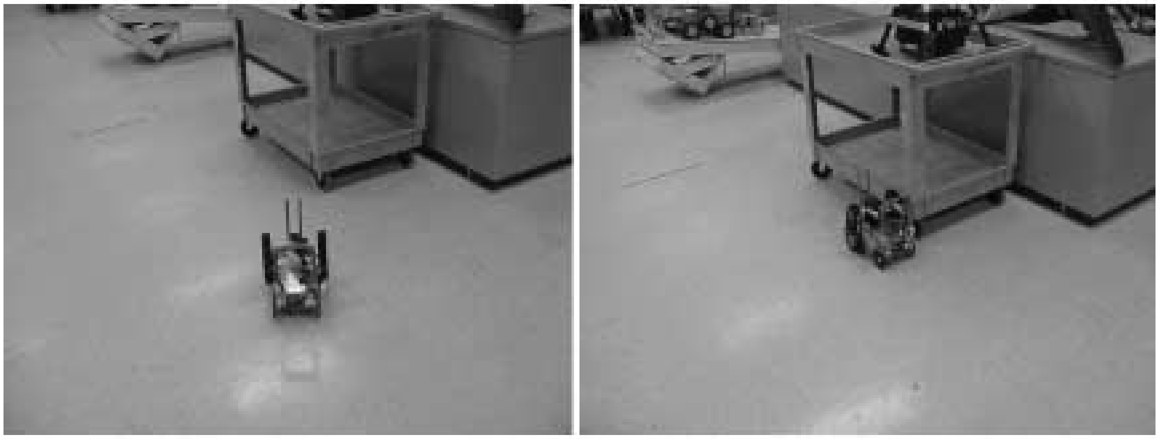

Figure 4-5: The robot avoiding an obstacle in the middle of the lab. The robot approaches the cart from the side, as shown in the left figure, and the robot is able to avoid the obstacle, as shown in the right figure.

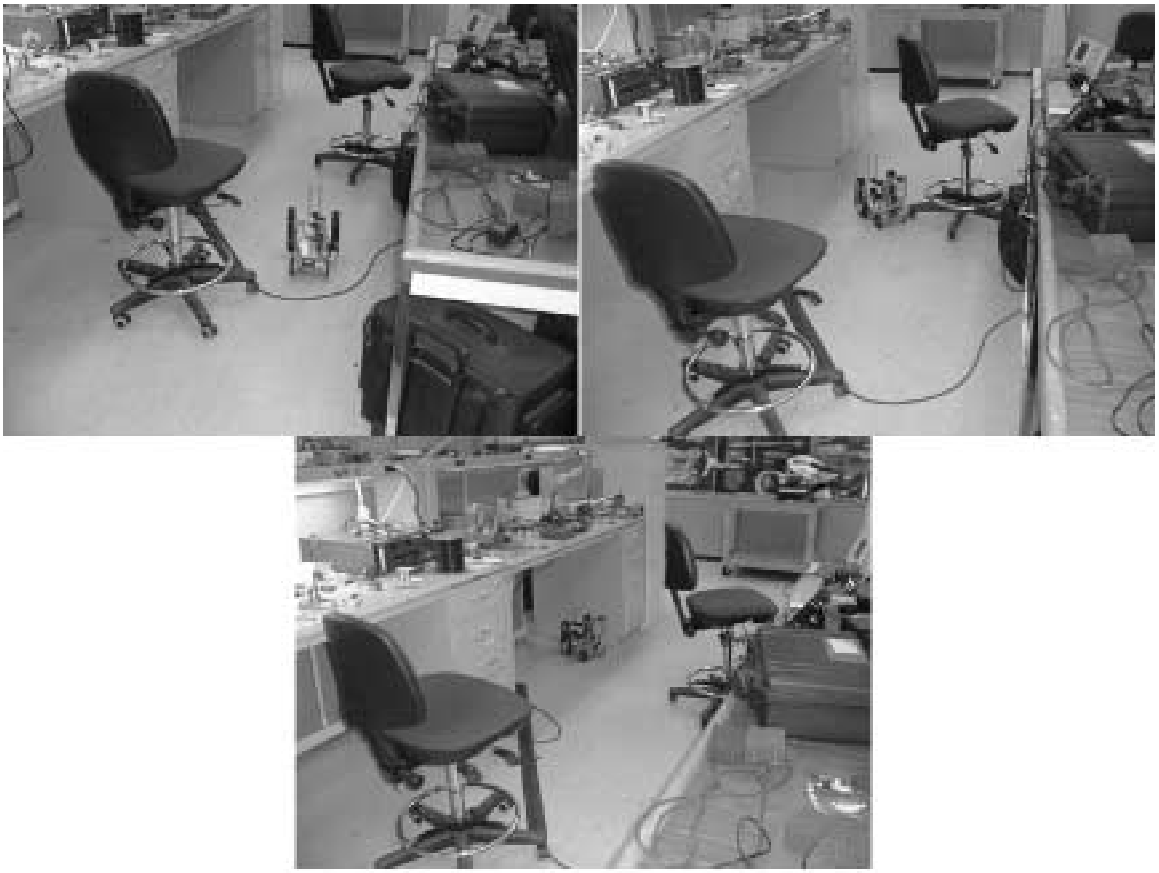

Figure 4-6: The robot avoiding an office chair. The robot approaches a chair (top-left figure) and avoids it by turning left (top-right). The robot continues going forward until encountering the desk drawers, where it avoids it by turning left (bottom). The robot is now caught in a dead end under the desk.

optical flow information on that side of the camera while the metal table leg gave significantly more flow on the other side.

The robot did not successfully traverse a passageway with lightly textured objects and walls on the sides. Figure 4-7 shows the robot unsuccessfully traveling down a hallway. In this case, the assumed optimization of computing the magnitude of flow by using only the horizontal component does not improve performance; the magnitude of horizontal flow of the black border of the hallway grows too slowly.

When traveling between a table and some cabinets, the robot collided into the smooth beige metal doors of cabinets (Figure 4-8). Collision with a wall also occurred when the robot tried to navigate the passage between tan-colored office cubicle dividers. The necessity of textured obstacles is confirmed when the robot is able to avoid colliding into the cabinet only after a textured obstacle is placed in the path.

For the obstacle avoidance to work, an obstacle needs to be off to the side of the robot's field of view. Figure 4-9 shows the robot both successfully and unsuccessfully driving through a doorway. The robot is unsuccessful when the edge of the doorway is centered in the robot's field of view.

Before the robot was able to successfully avoid obstacles, the turn coefficient multipliers drive correction range values, and other thresholds needed to be fine-tuned. The parameters also needed to be tested for each different patch size tested. Because an accurate model was not made for the dynamics of the robot, the drive correction parameters and thresholds were tuned through test runs.

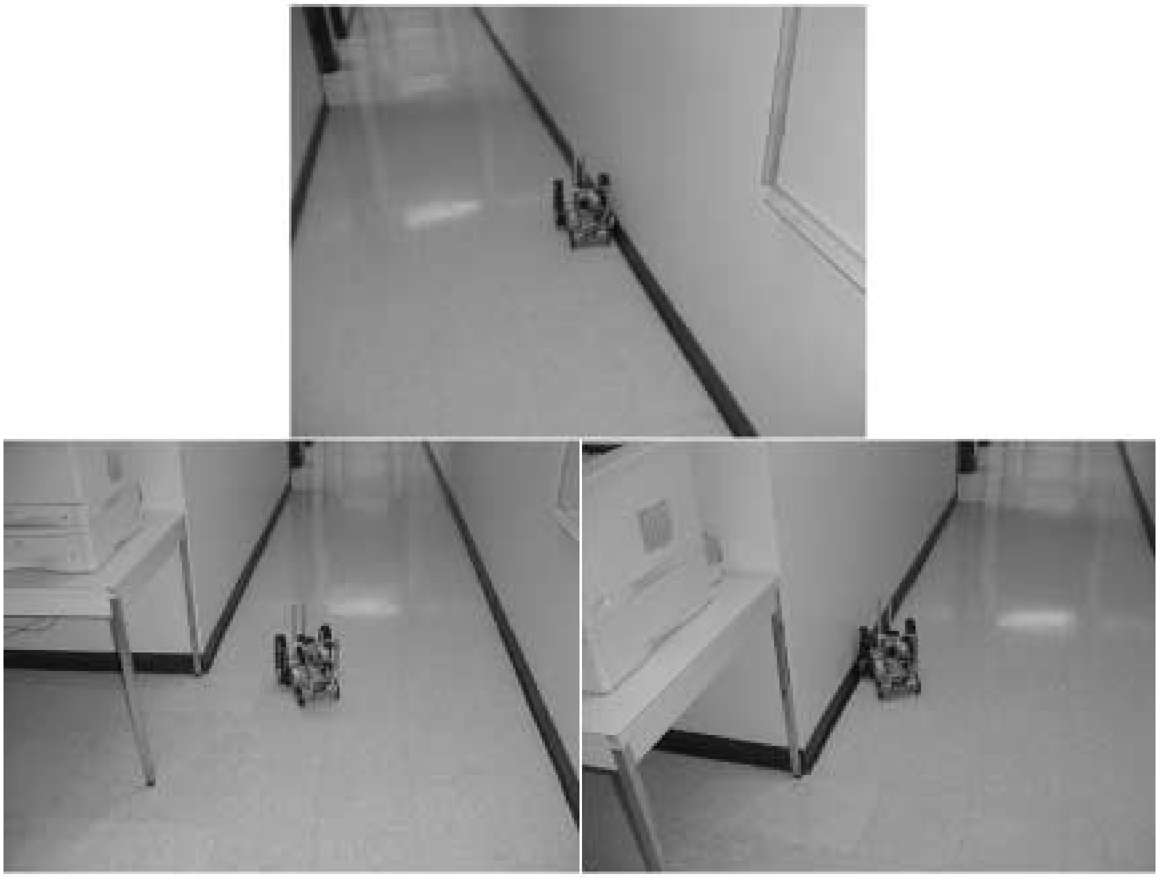

Figure 4-7: The robot traveling down a hallway. The upper figure shows the robot trying to drive along the right wall after turning left to avoid collision, but the robot is too close to the wall, and thus friction is preventing forward progress. The bottom two figures show the robot unsuccessfully avoiding the left wall. The bottom-left figure shows the starting position of the robot, and the bottom-right figure shows the robot colliding into the wall.

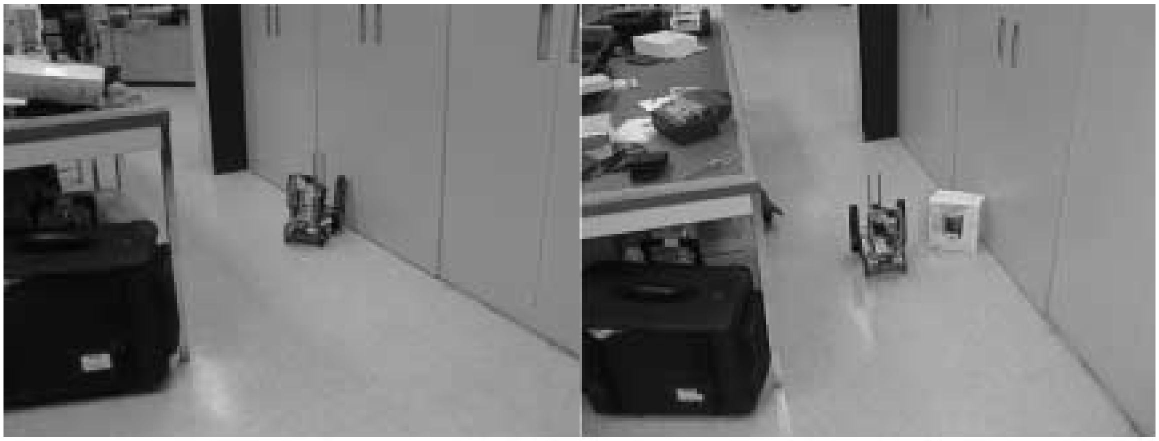

Figure 4-8: The robot driving between a table and cabinet. On the left, the robot collides into the lightly textured cabinet. On the right, the robot is able to avoid the cabinet after an additional textured obstacle is placed in its way.

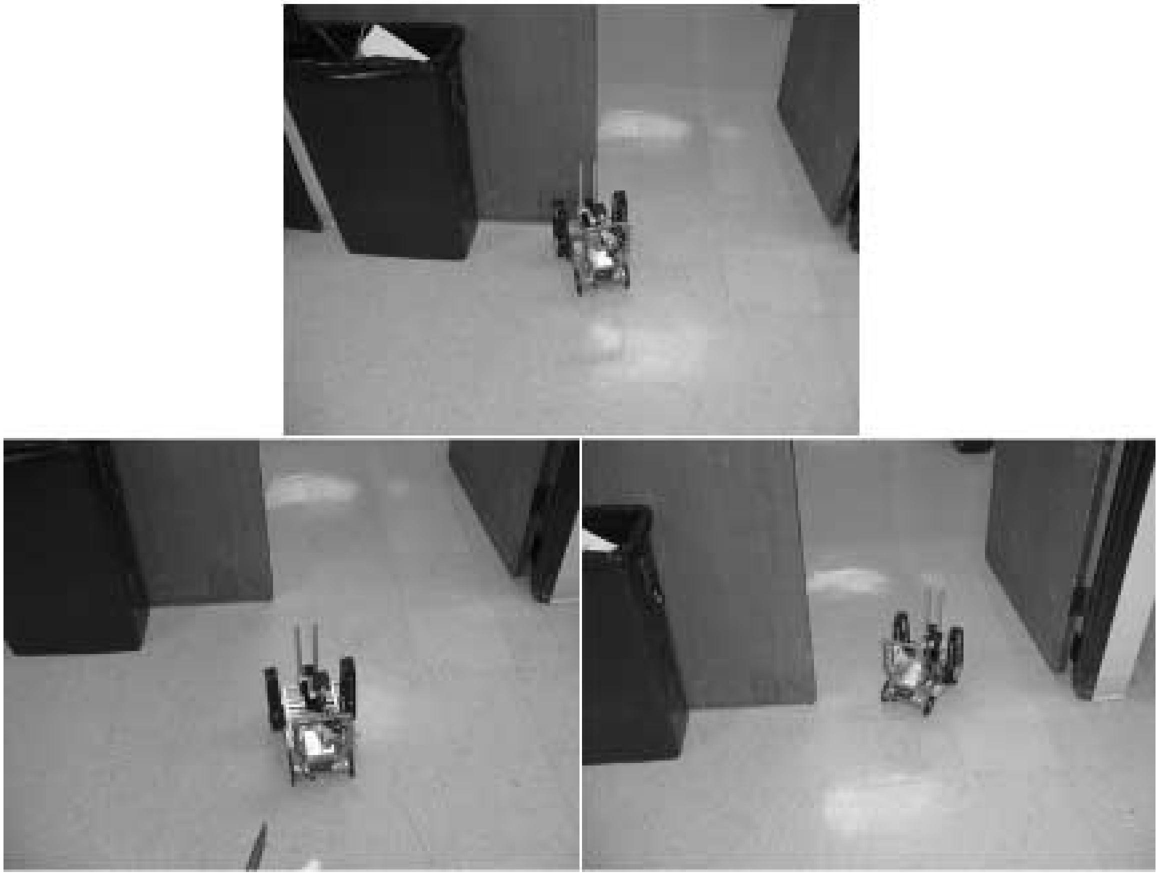

Figure 4-9: The robot driving through a doorway. When the edge of the doorway is centered in the robot's field of view, the robot is unsuccessful in avoiding it (top figure). When the robot drives forward with the edge of the doorway off to one side of its field of view (bottom-left), the robot is able to avoid it (bottom-right). In this case, the robot then is unable to avoid colliding into the door.

Chapter 5

Future Work

Future improvements of the system can not only expand the work done using optical flow for obstacle avoidance, but also new behaviors and better models can be designed and applied.

5.1 Improvements to Current Optical Flow Obstacle Avoidance

Firstly, different methods of computing optical flow can be implemented and tested instead of patch matching. The more robust algorithms which capture more global motion should be considered first. A multi-resolution approach coupled with an optimized local flow algorithm seems like the next best method to implement. Other dynamic reactive systems can be implemented in place of the optical flow balance strategy. The similar free-space local search method implemented by Horswill is a good first candidate [5].

A helpful time-to-collision measurement can be deduced. Because the incremental motion of the robot can be considered as a small translation, for each successive pair of video frames, the distance between objects can be computed. Purely translational motion allows for easy calculation of relative distances between objects using optical flow [13]. Knowing the distance to objects allows for improved obstacle avoidance.

The payload camera is mounted on a platform higher than the viewpoint of the robot's body. By lowering the camera even more, to match the level of the bow camera, the obstacle avoidance can detect flow of obstacles more directly in the way of the robot. Also, switching to a camera with a wider field of view can better detect obstacles.

The correcting drive commands were programmed without a model for the dynamics of the robot. Building a non-holonomic dynamic model of the robot would allow for better drive control correction by the obstacle avoidance module.

Testing of the current system was done only in an indoor lab environment. Future testing can be done in more varied indoor environments and also outdoor environments. Less rugged outdoor terrain, such as pavement and grass, can be tested first before possibly testing in as rugged of terrain as the small robot can handle.

Specific to implementation on the Trimeleon, the system should be programmed and run using an operating system and a multithreaded environment. Parallel processing would greatly speed up computation of optical flow and lead to better and more accurate performance.

5.2 Utilizing Full Robot Suite

This project is the first to utilize the payload port for interfacing additional sensors and processors to the HMTM robots for adding new behaviors and capabilities. The implemented payload module demonstrates the feasibility of building new features on top of the basic HMTM platform. Thus, future payload modules can be designed using the communication infrastructure as described in this project.

The image processing done thus far works only with the intensity values. The onboard and payload cameras do also capture color data. Thus color can be used beneficially for future image processing. For example, object boundaries can be defined by both intensity and color boundaries, which can make obstacle detection more accurate.

Also, the additional sensors onboard the robot can be utilized with the camera for a sensor fusion approach to navigation. Although the inertial sensors used are cheap and lose accuracy over time as the robot moves, coupling the inertial sensor data with the camera data can produce more accurate data from which to make navigation decisions.

5.3 Adding New Behaviors

New behaviors can be built on top of the existing capabilities of the robot. The visual servoing task for the payload module was conceived for an obstacle-free environment. Potentially, the visual servoing capability can subsume the obstacle avoidance module and thus handle its task in an obstacle-filled environment. By following the principle of a behavior-based subsumption architecture, the robot can evolve new capabilities by adding any series of incrementally more complex behaviors to the system [14]. As demonstrated through this project, the payload port makes adding higher-level behaviors possible, and thus can be utilized again in the future.

5.4 Building Better Environment Models

The purpose of creating a simple reactive obstacle avoidance system was that the system could be run onboard the robot in real time and also that the system was not constrained to any target environment. Even though optimizations were made to the system because the target environment was known, such as from the flat ground plane assumption, the goal of producing robots that can handle any environment is desirable. A general system will be able to be deployed in a wider variety of missions. Because only one general platform must be built, the cost of producing one system is lower than building many different ones for different tasks.

For a dynamic reactive system to work, some recognition of archetypal landmarks should be programmed into the system. Different than incrementally map building for the entire environment thus traveled by the robot, and different than preprogramming a specific map for the target domain for the robot, archetypal landmarks should be general symbols for objects. A doorway is an example for a general symbol for something potentially encountered by a robot, as are humans and tables and chairs. The robot can run dynamically by searching only the local environment for symbolic landmarks.

Finally, related to building a more general system, some of the constraining assumptions about the environment can be relaxed as a future improvement. The flat ground plane assumption for example is relied upon for optimizing optical flow patch matching. For outdoor environments however, the flat ground plane assumption will not hold. As the system becomes more able to handle more general environments, more of the underlying assumptions made about any target environment must be relaxed.

Chapter 6

Conclusion

An optical flow based obstacle avoidance system is designed for a small mobile reconnaissance robot as the first step in developing a complete semi-autonomous navigation system. The purpose of the system is to enable semi-autonomous behaviors in the robot such that the operator can spend less time doing low-level driving and instead spend more time on making more important high-level decisions. The system will also reduce operator mishaps while driving the robot.

The obstacle avoidance module relies on a simple optical flow balance strategy. Simply put, the balance strategy interprets an imbalance in the total magnitude of optical flow on either side of an image as an impending collision with a nearby obstacle; when flow imbalance is detected, the system signals the robot to turn and avoid the likely obstacle. The system uses a fast patch-matching optical flow computation algorithm. Optimizations for improving algorithm performance include using logarithmic search for patch matching, scaling down image size, and balancing image differences to combat image noise.

The obstacle avoidance system is implemented on a payload module interfacing with the host vehicle. Leveraging the existing communication protocol between robots and operator control units, the payload module is able to manipulate OCU and robot communication data in order to issue drive corrections to its host vehicle in order to avoid obstacles.

The system was first implemented and tested within a virtual environment by using an open source 3-D graphics engine. The physical system was then tested indoors in a lab environment. Both virtual and real world robots were consistently capable of avoiding nearby obstacles and traveling down corridors. Both testing environments also exhibited the weaknesses of the system: incorrectly avoiding shadows as obstacles, colliding into textureless objects, and running into obstacles centered in the field of view. Overall, the system performed well in correcting operator drive commands to avoid obstacles, and as well as expected under expected failure modes.

Suggested future work with the system should explore adding new behaviors to the robot in a subsumptive architectural manner. The production of a reactive dynamic navigation system should still be the main goal, which can be aided by developing archetypal symbols for recognizing objects in any general environment.

Bibliography

[1] Terrence W Fong and Charles Thorpe. Vehicle teleoperation interfaces. *Autonomous Robots*, 11(1):09–18, July 2001.

[2] Global Hawk. http://www.af.mil/news/factsheets/global.html.

[3] RQ-1 Predator Unmanned Aerial Vehicle. http://www.af.mil/news/factsheets/RQ_1 _Predator_Unmanned_Aerial.html.

[4] iRobot Corporation Packbot. http://www.irobot.com/rd/p08_PackBot.asp.

[5] Ian D. Horswill. *Specialization of Perceptual Processes.* PhD thesis, Massachusetts Institute of Technology, 1993.

[6] Liana M. Lorigo. Visually-guided obstacle avoidance in unstructured environments. Master's thesis, Massachusetts Institute of Technology, 1996.

[7] Berthold K. P. Horn. *Robot Vision.* The MIT Press, 1986.

[8] S. A. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Trans. Robotics and Automation*, 12(5):651–670, October 1996.

[9] B. Horn and B. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[10] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[11] Yao Wang, Jorn Ostermann, and Ya-Qin Zhang. *Video Processing and Communications.* Prentice Hall, 2002.

[12] Crystal Space 3D. http://crystal.sourceforge.net.

[13] Kurt A. Steinkraus. Optical flow for obstacle detection in mobile robots. Master's thesis, Massachusetts Institute of Technology, 2001.

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