I See You!

Robotic Surveillance Using Appearance-Based Obstacle Detection

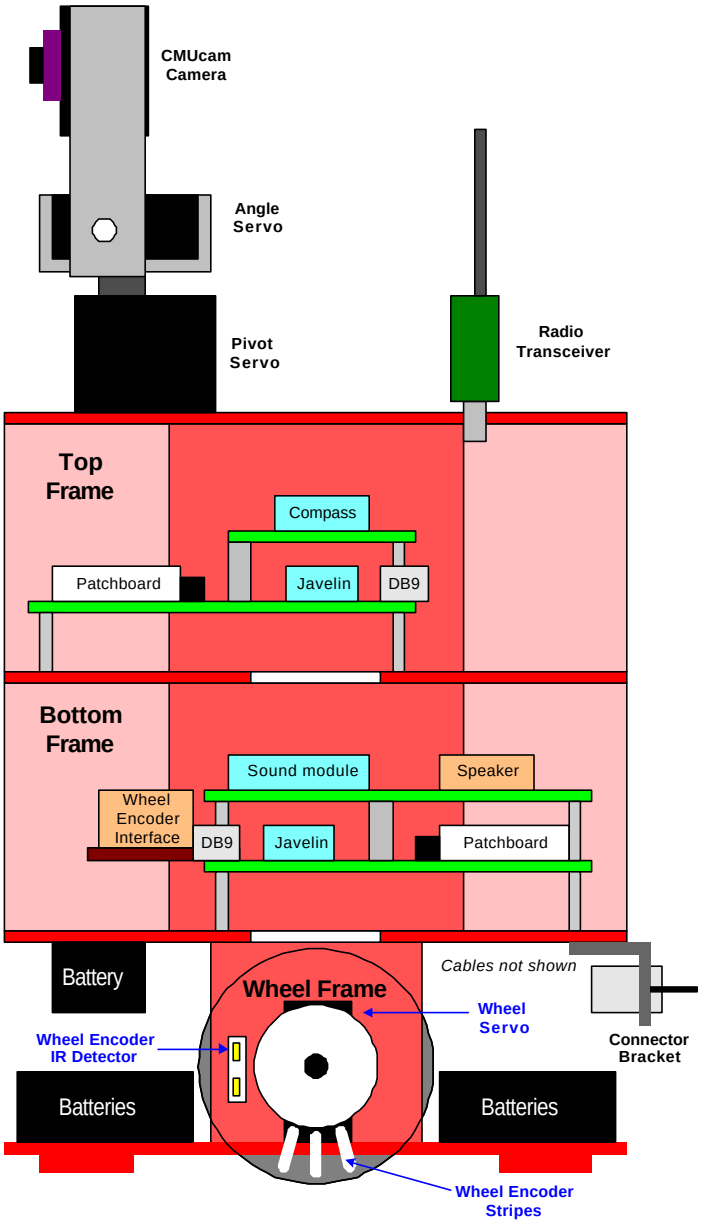

Table of Contents

List of Figures

List of Photographs

List of Tables

Code Listings

1. Problem

Robots can augment or replace humans in hazardous areas such as military conflicts, fires and underwater. To do this a robot requires knowledge of its surroundings. Using a variety of sensors increases the complexity of the robot. A suitable vision-based system should be able to replace a variety of sensors to provide information about surrounding obstacles.

2. Hypothesis

It should be possible to construct a vision-based, obstacle detection and surveillance system that will work with a small, autonomous, self-contained, battery-powered mobile robot. A single CMUcam camera should provide suitable data for identification of other robots and obstacles. This assumes that obstacles will have colors that are distinct from the floor color, obstacles will be at least six inches on a side and the floor will be flat.

3. Abstract

People use their eye's everyday to locate objects and avoid obstacles by their appearance. A robot can use simple obstacle detection systems such as range finders, but a camera based system makes it possible for a robot to employ methods similar to those used by people. This project shows how simple image analysis, such as the mean color of an area, can provide sufficient navigation feedback for avoiding obstacles and locating objects under surveillance by their appearance. This more closely emulates the way people avoid obstacles.

The project employs a battery-operated, mobile robot with two Java processors coupled with a movable CMUcam camera. The robot detects objects using basic image processing and the camera's ability to report and track blocks of colors. No other obstacle sensors were employed. Up to three robots were used in various experiments to visually locate fixed and moving objects within a room. A radio transceiver on each robot allows for cooperative searches. The environment was controlled to allow the robots free movement. This included a flat surface, even florescent lighting and objects that could be visually distinguished.

The CMUcam was tested using a variety of colored sheets and floors to determine a suitable environment. No overhanging obstacles were allowed. Although the CMUcam generated RGB (red, green and blue) data, experiments showed that conversion to HSI (hue, saturation, intensity) made obstacle recognition easier. It also minimized the affects of lighting and shadows.

Experimental results show that full image analysis is not necessary for obstacle avoidance and object location. The output from the camera was analyzed in a grid to

identify objects. Finer grids provided more accurate obstacle determination. The CMUcam's low 80 x 143 pixel resolution turned out to be suitable for the obstacle recognition task although this did limit the accuracy of the results.

The robots were able to navigate and location other robots and obstacles only using visual means. The large area observed by the camera made searching faster than would be possible using other sensors and allowed object recognition by color in addition to location.

4. Introduction

This project was inspired by work presented in the paper, *Appearance-Based Obstacle Detection with Monocular Color Vision* (Nourbaksh and Ulrich, 2000) by Nourbaksh and Ulrich. The system employed a single camera for obstacle detection for in conjunction with a robotic navigation system. The camera and supporting microcomputer were used with a large mobile robot. I wanted a similar system that would work with a small robot. The solution would have to be considerably less powerful, lower in weight and power requirements. This means the performance of my alternative would be lower but it should be more powerful than other alternative obstacle detection systems because it can also detect the color of an object. It is assumed that reasonable restrictions can be placed on the robot's environment including limiting the area of navigation to a flat surface and choosing obstacles with colors that can be differentiated using the chosen camera.

This project was designed to create and evaluate a low cost, obstacle detection system suitable for use with a low cost mobile robot such as the Parallax Javelin Bot. The system will be successful if it provides sufficient information to the robot for navigation around obstacles and recognition of specific obstacles such as other robots. A color camera, the CMUcam, was chosen as the sensor for the project. Obstacles were to have distinctive colors allowing them to be differentiated from each other. This sidesteps the issue of camouflage that will probably be beyond the capabilities of the CMUcam and the Javelin Stamp microcontroller used on the test robot.

The project is to determine if the CMUcam/color differentiation approach is feasible. This turned out to be true and the subsequent work is related to the system's use

in obstacle detection and recognition for surveillance purposes. The project will also determine how accurate the system can be. Problems due to the limitations of the CMUcam were expected based on comments found in the paper *Illumination and its effect on CMUcam* (Rowe, 2001b).

A better obstacle detection and recognition system is needed to address the final part of the project that is to support a basic robot surveillance system.

4.1 Plan of Action

A good deal of experimentation is required before a robot can actually use the CMUcam for surveillance purposes. The first step is to profile the camera so its capabilities and limitations can be taken into account. The documentation that comes with the camera provides only a starting point for this work. The next step is to determine a methodology for identifying obstacles and their position relative to the robot. Finally, the system is used with a robot to perform basic navigation and surveillance.

5. Research

This project was inspired by work presented in the paper, *Appearance-Based Obstacle Detection with Monocular Color Vision* (Nourbaksh and Ulrich, 2000) by Nourbaksh and Ulrich. The system employed a single camera for obstacle detection for in conjunction with a robotic navigation system. The camera and supporting microcomputer were used with a large mobile robot. I wanted a similar system that would work with a small robot. The solution would have to be considerably less powerful, lower in weight and power requirements. This means the performance of my alternative would be lower but it should be more powerful than other alternative obstacle detection systems because it can also detect the color of an object. It is assumed that reasonable restrictions can be placed on the robot's environment including limiting the area of navigation to a flat surface and choosing obstacles with colors that can be differentiated using the chosen camera.

I have been working with small robots for a number of years and sensor limitations have always been a problem. Infrared and ultrasonic range finders provide accurate distance results but it is difficult to locate individual objects (Wong, 2002). They also require multiple sensors or mechanical means to move the sensors to cover a wide area. The sensors only provide range information. They do not provide any other information about objects such as height and color.

A video-based sensor system has the potential to provide range information in addition to other object attributes such as color and height. One approach is to use two cameras for binocular vision but this has a number of problems including high complexity, difficult calibration and it requires powerful and expensive processing

resources. The system presented here employs a monocular vision system with a single camera, the CMUcam.

The CMUcam is a low cost camera created to perform a task of tracking a blob of color at 16.7 frames per second (Rowe, 2001a). It uses a CMOS color camera module and an Ubicom SX microcontroller that provides basic image analysis eliminating the need for a separate frame buffer although the system can download a screen dump. It is designed for use on small mobile robots via a serial connection.

5.1 Purpose

This project was designed to create and evaluate a low cost, obstacle detection system suitable for use with a low cost mobile robot such as the Parallax Javelin Bot. The system will be successful if it provides sufficient information to the robot for navigation around obstacles and recognition of specific obstacles such as other robots. A color camera, the CMUcam, was chosen as the sensor for the project. Obstacles were to have distinctive colors allowing them to be differentiated from each other. This sidesteps the issue of camouflage that will probably be beyond the capabilities of the CMUcam and the Javelin Stamp microcontroller used on the test robot.

The project is to determine if the CMUcam/color differentiation approach is feasible. This turned out to be true and the subsequent work is related to the system's use in obstacle detection and recognition for surveillance purposes. The project will also determine how accurate the system can be. Problems due to the limitations of the CMUcam were expected based on comments found in the paper *Illumination and its effect on CMUcam* (Rowe, 2001b).

A better obstacle detection and recognition system is needed to address the final part of the project that is to support a basic robot surveillance system.

5.2 Literature Review

Video-based obstacle detection normally uses one or two (stereo) cameras. They can be divided into laser-based and scene analysis approaches. Laser-based systems utilize a laser to illuminate part of the viewing area covered by a camera. The attributes and position of the reflected laser light provide obstacle location information. These systems tend to be limited to delivering location information. Implementation tends to be straightforward.

Scene analysis takes an image generated by the ambient light. This is the same type of approach used by a person looking at a scene. Scene analysis is often difficult because of the amount of information that must be processed, but it can provide information that a laser-based system cannot provide such as color and texture information that can be useful in identifying objects. Infrared cameras can provide temperature information, but they lack the ability to provide color information.

Robot surveillance systems must normally identify an object so the robot can differentiate it from other objects. Tracking a rock or rabbit is not useful if the desired target is a person or a vehicle. This makes scene analysis a desirable tool for surveillance purposes.

Laser-based obstacle detection systems tend to be complicated. They also require significant amounts of processing power to analyze laser data and calibration can be difficult. In the paper *Laser-Based Obstacle Detection and Avoidance System* (Abbot et al., 2000), the Center for Self-Organizing and Intelligent Systems' (CSOIS), Rocky

Rover utilized five fixed planar lasers and two CCD cameras. Positioning of the lasers and cameras was critical. The cameras were only used to record information on reflected laser light. Obstacles were detected by determining where laser light line bends occurred in the camera image.

The system provides very accurate position and distance information necessary for navigating in tight quarters. Unfortunately there were some problems in addition to the added weight of the sensor system. For example, some obstacles absorbed the laser light instead of reflecting it back preventing the system from providing range information about the object.

Another problem was due to diffusion. Diffusion occurs when the object disperses the laser light before being reflected. This can result in pixel loss or multiple reflections that caused successive errors in the pixel calculations. This experiment used fuzzy mapping. Fuzzy mapping represents two variables and provides a local memory of the area around the robot. Although the mapping system worked well for most environments, it did have problems with absorption and diffusion.

In *A Laser Range Scanner Designed For Minimum Calibration Complexity* (Chen and Davis, 2001), the problem of complex calibration was addressed with a system consisting of a single camera and laser. A four-mirror system provides a split view of an object to a single camera. This cuts the number of cameras in half but adds the complexity of the mirror system. In practice, the mirror system was easier to build and calibrate. Unfortunately the system tends to be a bit cumbersome if it needs to be tilted or rotated.

Laser sensors can be used on vehicles to look at each angle of an area and measure the range and intensity of reflected laser light back to a computer (Hancock, 1999). This approach is desirable for vehicles that often used at night because lasers are best used at nighttime where they provide different information then regular video data. Histogram analysis can be used to pick out possible obstacles that are not supposed to be in that area. Current laser systems are not capable of keeping up with the speed of a car moving towards obstacles, but improved lasers and higher performance computers may allow this type of obstacle avoidance system to be used on highway in the near future. Unfortunately, the cost of these lasers is high compared to a small robot.

Cameras are employed in robot-based surveillance systems but they often the cameras are use to send video information back to a remote site (Trebi-Ollennu and Dolan, 1999). Utilizing a camera for automatic surveillance as well as obstacle avoidance and navigation is less common. This is because scene analysis is a difficult problem.

Using two cameras operating in stereo system is a common approach scene analysis. A stereo vision system is computationally expensive because it must find corresponding pixels in both images. A stereo vision system also requires two cameras that requires more memory and power. Attempts have been made to improve the performance of stereo vision systems (Hirschmüller, 2001), but they still require more processing power and memory than is found in most low cost robots.

There seems to be very little research involving single camera, scene analysis vision systems. One notable paper by Nourbaksh and Ulrich (Nourbaksh and Ulrich, 2000) describes a system that used only a single, high resolution camera with a PC to provide obstacle identification for real-time navigation support. It uses a color-based

approach that is similar but more complex than the one used in this project. It assumes that each individual pixel can be placed into one of two categories: ground or obstacle. It also assumes that the color of the ground is relatively consistent. Obstacles can then be located by the difference in color between the ground and obstacle.

The difference was determined by converting the RGB (red, green, blue) camera information to HSI (hue, saturation, intensity) information and then using the hue and saturation data to differentiate obstacles from ground. This minimized the affect of ambient lighting and provided very good obstacle recognition results. The approach has the advantage of being simple to implement and calibrate since the system involves only a single camera.

The ground color is obtained by looking at an area in the image that corresponds to the an area immediately in front of the robot. The assumption is that the robot starts without an object immediately in front. Various approaches were tested starting with a single floor sample was taken when the robot starts to subsequent samples while the robot moves.

Using a vision system for robot navigation is only one possible use. Surveillance is another and often the two can be combined. Using multiple robots can provide improved system performance by combining the information obtained from the group. Distributive surveillance systems allow robots to see what is occurring around them (Diehl et al., 2000). This work evaluated images to try to determine the kinds actions objects within view were doing. For example, a person may be walking or waving to another person. Classification of these motions was possible even though the resolution

of the video system was low because multiple images were obtained from different reference points.

The authors recommended using a distributive surveillance system that did not limit the capabilities of the sensors. The distributed approach also increases system coverage and improving fault tolerance and while lowering the cost.

Most vision-based obstacle detection and image analysis research has been oriented towards systems with high computation and memory facilities. Often, scaling these methods to a lower performance system is difficult or impossible because the methods have requirements such as a minimum image resolution to be practical. What is necessary is a ground up approach that takes general ideas such as the use of HSI data (Nourbaksh and Ulrich, 2000).

In conclusion, laser systems appear to have high costs, performance and weight requirements making them unsuitable for a small mobile robot. Stereo visual systems suffer the same problem especially because of the complicated programming involved. A single camera visual system appears to meet the cost, weight and performance requirements if the results provided by such a system are suitable for navigation and obstacle detection.

6. Materials And Construction

The EBot, as it has been named, is a custom mobile robot that is equipped with a CMUcam for monocular vision to provide obstacle detection. Three identical robots were built. Eventually they will be used for more sophisticated surveillance experiments involving multiple robots.

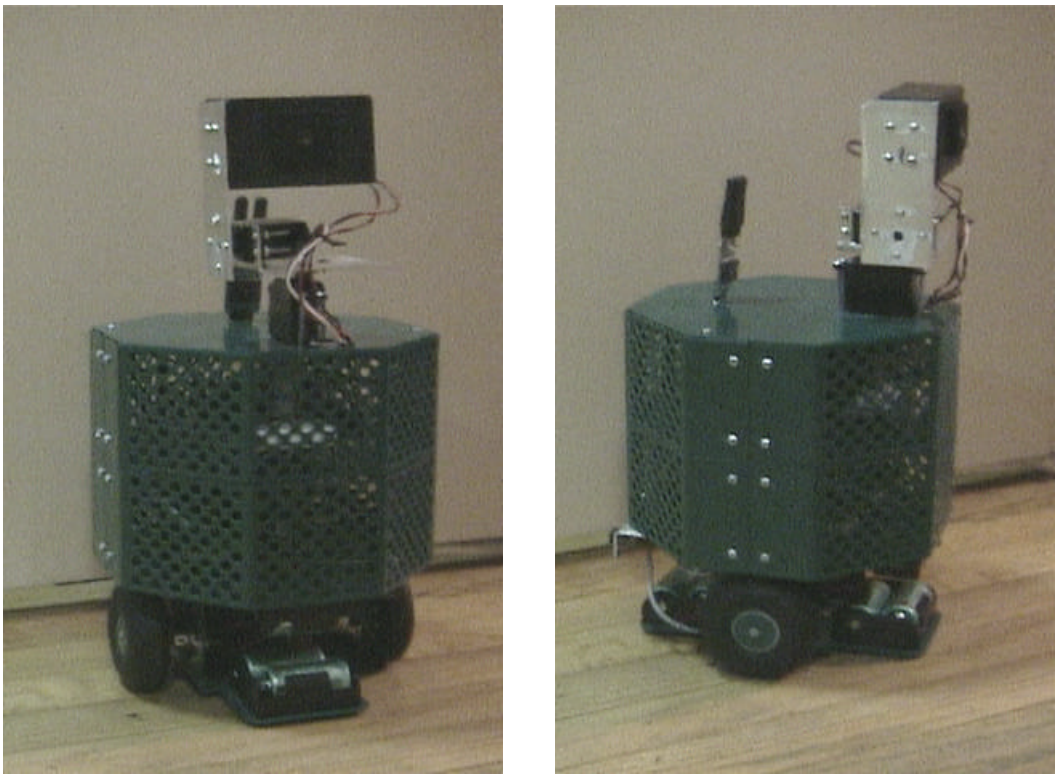

Photo 1: Ebot Side and Front

The frame of the robot is made out of two metal octagonal shapes. These were obtained from Parallax and Michael Berta Enterprises (www.easybots.net). The trade name for the frame is EasyBot. The frames were modified to handle the various peripherals and sensors including the CMUcam camera, radio transceiver and wheel encoder sensors. The next figure shows a diagram of the EBot.

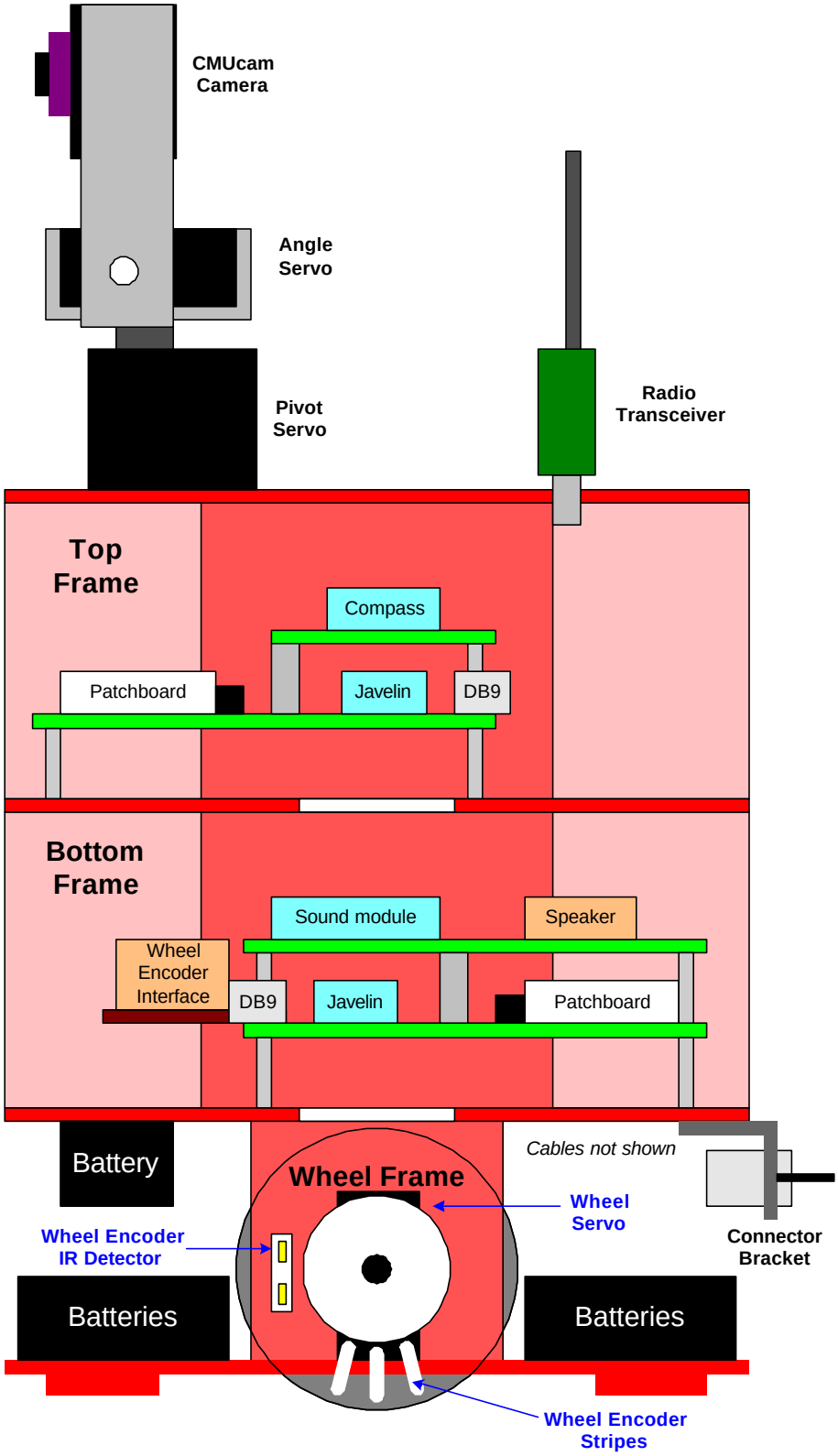

Figure 1 : Side View Of Robot

The CMUcam is the only obstacle sensor used with the EBot. Additional sensors may be added in the future but this project is designed to prove that the camera alone is sufficient for sophisticated navigation. The CMUcam was developed at Carnegie Mellon University and is available from Seattle Robotics (www.seattlerobotics.com). It is also available from other sources.

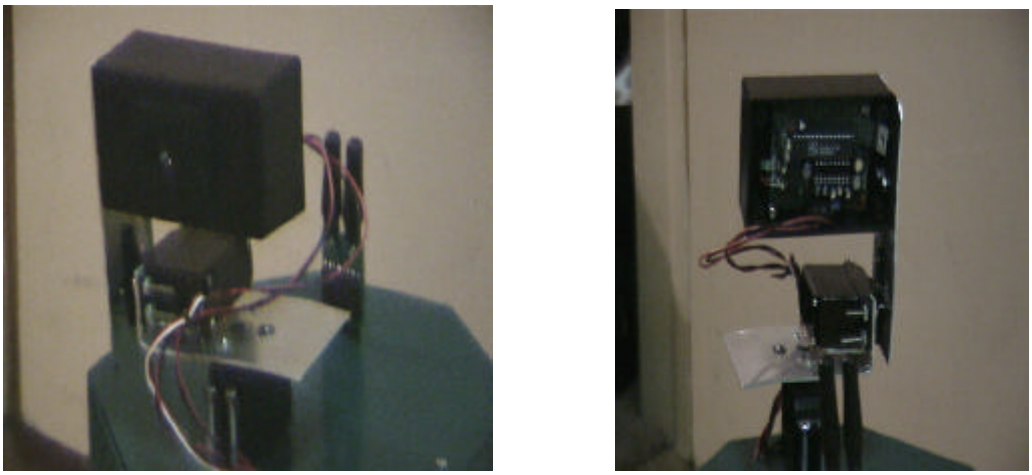

Photo 2: CMUcam and Servo Mount

The CMUcam was mounted in a plastic box obtained from Radio Shack. The servo/camera mounting brackets shown above were custom made from 1/8” aluminum.

The robot includes two Javelin Stamp boards, a radio transceiver, a compass module, a sound module and speaker, a wheel encoder interface, and four separate servos. Two of the servos are for the camera allowing it to pivot and tilt. The other two servos for moving the robot. They are modified so they rotate continuously and they are attached to the EBot’s wheels.

The Javelin Stamp boards are where all the programs from the PC are stored. There are two Javelin Stamp boards per robot to distribute processing jobs, provide

support for a larger number of peripherals and increase the amount of RAM available for applications. Each board contains a Javelin Stamp that runs a limited version of Java. It contains 32kbytes of memory. The top system controls the camera servos and camera. It also controls the compass module and radio transceiver. It is linked to the bottom board using a full duplex serial link. The upper unit tells the lower what to do with the wheels and soundboard. The lower unit uses wheel encoder sensors to track the wheel movement to keep track of the robot's relative movement.

The radio transceiver is a radio frequency module that sends and receives information from the PC. It can receive signals as far away as 150 feet from a PC but in these experiments the robot will travel the maximum of twenty feet away from the PC. The radio transceiver is used to send status information to the PC.

Parallax Compass Module plugs into BOE (Board of Education) module socket. The compass module intuitively depicts eight directions with just four LED's but this information is accessed via serial interface. The serial interface is used to determine the orientation of the EBot. Prior experiments have determined that the EBot will be able to keep track of its position very accurately using the compass and the wheel encoder information.

The Compass Module is not very precise and it is very sensitive to external electromagnetic sources such as the servos. For this reason, the unit was mounted above the EBot away from the wheel servos. Although the compass is not very precise it is reliable within its limited precision.

The soundboard is the Parallax BOE-Bot Speech Board with the Diphone Chip Set installed. It allows the robot to use speech and sound for feedback. The diphone support allows speech generation using phonemes.

The wheel encoder support utilizes two Fairchild QRB1113 Infrared Reflective Object Sensors to track wheel movement. The white strips were painted on the inside of the wheels inline with the wheel encoder sensors. The sensors can detect the transitions between white paint and the black wheels. It was not possible to use wheel encoder labels because they could not be attached to the inside of the wheels.

Photo 3: Ebot Mounting Bracket

A custom mounting bracket and connector interface, shown above, was built to so a single power and serial port interface could be used for both Javelin Stamp boards. The robots run on five C nickel metal hydride (NmH) batteries. An extra battery holder was added to the four that come with the EasyBot frame. The photograph shown above also shows the pattern on the floor used for testing.

The following diagram shows how the modules, servos and Javelin Stamp boards were connected.

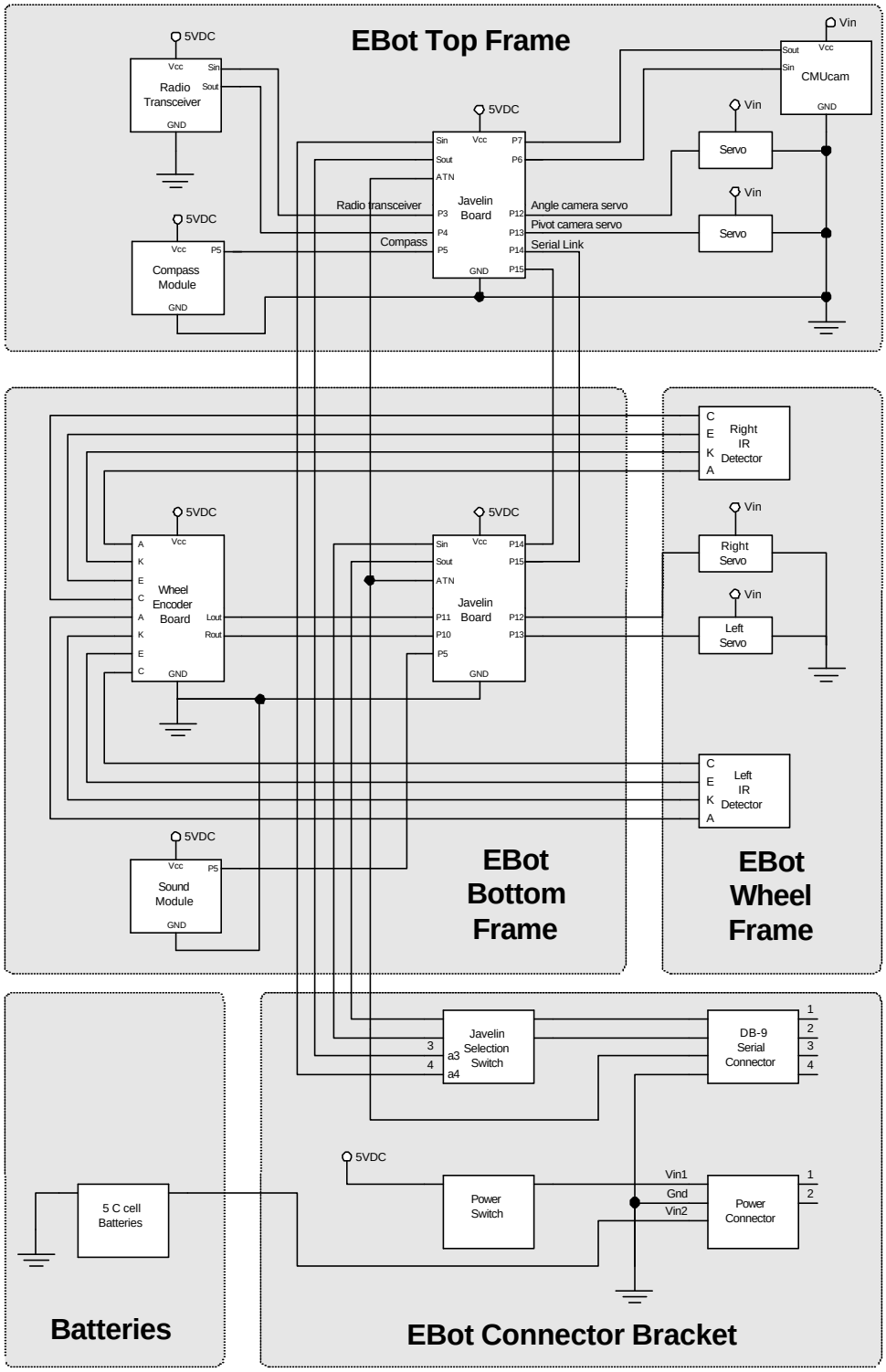

Figure 2: EBot Logic Diagram

The Javelin Stamp board has sixteen single bit ports that can be programmed for a variety of purposes including serial links. There are more than enough ports for the project but unfortunately there are other limits due to software. The Javelin Stamp utilizes “virtual peripherals” for devices such as serial ports and servo controls. It is limited to six active virtual peripherals. It is possible to turn off or deactivate a virtual peripheral and activate it at a later time.

7. Test Procedures

The goal of this project is to deploy a set of small, mobile robots that use a video camera to navigate through a room with a flat floor, avoiding obstacles and demonstrating the ability to locate obstacles and other robots by their color. This chapter describes the construction of the robots and camera followed by a description of the experiments performed to make this possible. These include:

1. Profile Gray and Robot Colors
2. Profile View To Floor Mapping
3. Obstacle Detection By Color
4. Pivot Angle And Camera Coverage
5. Static Obstacle Detection
6. Obstacle Detection Speed
7. Obstacle Detection While Moving

The EBot architecture is described in the first section. It uses some standard parts, such as the Parallax Javelin Stamp Board and add-on modules. Much of the remaining hardware was custom made including the servo mounts for the camera and the wheel encoder detectors. The robot will be used for future research, so all of its capabilities such as speech output and wireless communication are not exercised by these experiments.

The camera attached to each robot used in the experiments is the CMUcam. The CMUcam has the ability to compute the mean color values and the delta for each primary color (red, green and blue or RGB). This *color information* is the primary source of data recorded in various experiments and used by the programs that control the robots to

detect open space, obstacles and other robots. The color information is often converted to other color representations such as hue, saturation and intensity (HSI).

A range of experiments were necessary because the CMUcam does not support obstacle detection directly. It does have a microcontroller that provides an intelligent serial interface but the documentation and sample programs provided with the camera do not profile its capabilities beyond the serial interface protocol.

The Profile Gray and Robot Colors experiment provides a baseline for color recognition. The colors employed in the experiment are ones that will be used in subsequent experiments but with different characteristics such as different positions with respect to the camera. The size of colored obstacles also affects imaging results.

The Profile View To Floor Mapping experiment is necessary because the camera does not provide a one-to-one coordinate mapping between the actual area and the 2D view recorded by the camera. This is apparent when using the CMUcam sample application to view full frame results returned by the camera. The image is slightly distorted and the image is not centered with respect to the camera lens. The experiment determines the camera's characteristics so results from the camera can be translated to real world coordinates.

The Obstacle Detection By Color experiment tests the camera's ability to detect objects that do not fill the entire viewing area. It also shows how well the system will perform using various partitioning schemes.

The camera is mounted on a pivot and angle servo that provide the camera with two degrees of freedom. The Pivot Angle And Camera Coverage experiment determines

what positions are useful when the system is trying to view a large area around the robot. The results are needed to determine overlapping viewing regions.

The results of the prior experiments are used to create a program that can detect obstacles and report their position relative to the robot. The Static Obstacle Detection checks the accuracy and suitability of the program for use in subsequent experiments. Both the robot and all obstacles remain in fixed positions.

The obstacle detection system will be used with the robot's navigation program. The Obstacle Detection Speed experiment is needed to determine how fast the robot will be able to recognize and determine the position of obstacles so it can avoid them. The faster the detection speed, the faster the robot can move.

The Obstacle Detection While Moving experiment is the last of the group. This tests to see how well the robot can move within a room with obstacles, including the walls.

7.1 Profile Gray and Robot Colors

The CMUcam has limited color accuracy and range making it difficult to distinguish subtle color differences. This experiment determines how well the camera differentiates large areas of solid colors and what colors can be identified programmatically. This is done by presenting the camera with individual sheets of different colors and recording the camera output. Colors for obstacles and robots in subsequent experiments are chosen based on colors that can be readily identified and differentiated.

The experiment, as with subsequent experiments, utilizes the auto-white balance and automatic gain support of the CMUcam. This is done by placing a card that has a

19% gray color, also called neutral gray, about six inches in front of the camera lens so that it will fill the area viewed by the camera. The auto-white balance (AWB) and automatic gain control (AGC) are turned on for at least ten seconds and then turned off. This allows the camera to adjust to the room lighting so that subsequent readings will be more consistent. The AWB and AGC are disabled during experiments to prevent transient changes by the camera that will continually adjust based on lighting conditions that would make it difficult to reproduce experimental results. Future experiments may employ dynamic AWB and AGC support.

The color sheets must be large enough to cover the entire area viewable by the camera. The camera was aimed at the floor at an angle of 15º from horizontal to minimize the area. The floor color results were obtained by placing the robot and camera on the floor without any colored papers.

These tests were performed using a Java test application running on the top Javelin processor. It displays the mean color value for the entire CMUcam image capture area (80x 143 pixels). This was obtained using the GM command for the CMUcam serial interface. A single value was returned using poll mode. The information is of the form red, green, blue mean for the area and the variance for each mean value. This information must be copied from the Javelin debugger message screen into a spreadsheet.

Six colors were tested in this experiment; white, lime, yellow, pink, green, and blue. The test is repeated three times each for each colored sheet. The floor was as standard hardwood floor with slats that were three inches wide. The coloring varied from a dull yellow to a light brown. The average color within a five to six inch area was consistent through the room where testing occurred.

7.2 Profile View To Floor Mapping

The CMUcam does not provide a one-to-one coordinate mapping between the actual area and the 2D view recorded by the camera. This experiment is designed to determine the proper translation from the recorded data and a model representation of the original area. This procedure will be necessary to determine the real, relative position of obstacles with respect to the robot.

This experiment requires a PC running the Java test application that comes with the CMUcam. The CMUcam must be connected to the PC via the CMUcam's serial link which means the serial chip must be installed. This is normally removed when the CMUcam is working with the robot's Javelin Stamp microcontroller. The Java test application screen dump feature is used to display what the camera is viewing.

The experiment will determine two pieces of information. The first is the angle that the camera needs to be aimed to view the desired area. The angle is with respect to the horizontal plane that is parallel to the top of the robot as shown in the next figure.

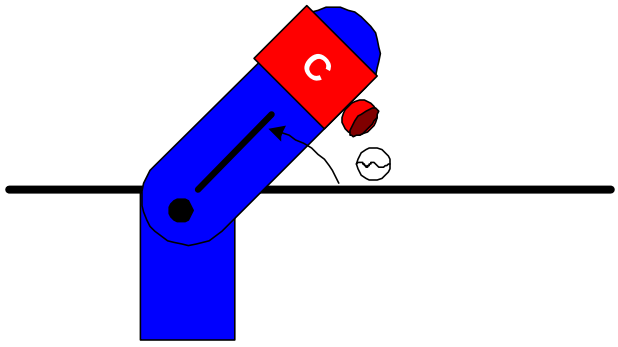

Figure 3: Camera Angle

The second piece of information is how the image is distorted. The next figure shows how this occurs.

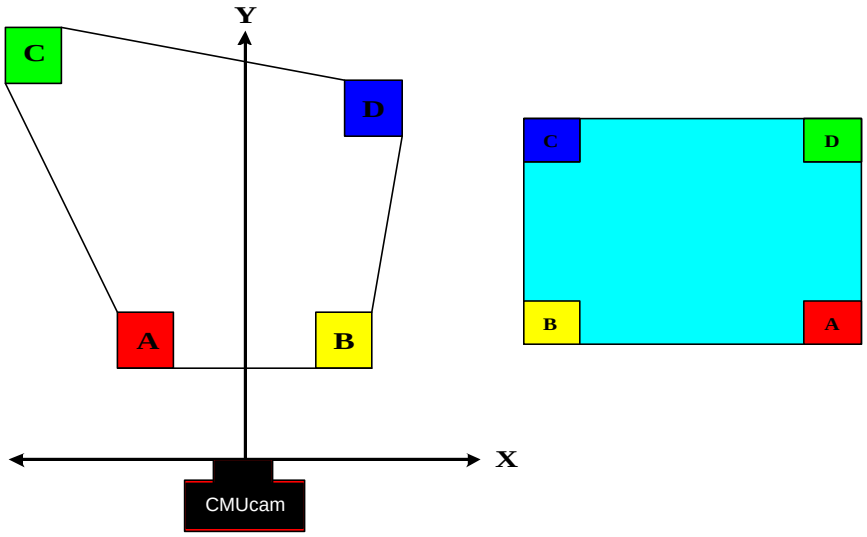

Figure 4: CMUcam Image Distortion

It was expected that some distortion would occur because the CMUcam is equipped with a wide view lens that tends to warp the image near the edges and corners. What was not expected was the more significant distortion that the camera has. The distortion is apparent when viewing objects using the screen dump facility. The figure above indicates how the image generally maps between the real area on the floor in front of the camera versus the screen. Note that the left to right translation is due to the Java application operation that flips the image. This type of compensation is easy because it involves a linear translation.

The two pieces of information must be obtained for multiple camera angles so that suitable coverage can be obtained. A low angle will allow the camera to see the area very close to the robot while higher camera angles will allow the camera to view a larger are or most of the room that it will explore. This experiment must determine whether one

or more viewing angles will be necessary to provide sufficient information about the surrounding area for navigation planning purposes.

The experiment requires four colored and/or numbered pieces of paper about two inches on a side that can be easily seen in a screen dump. The robot was placed on a floor with empty space surrounding it. The A and B markers are placed in the desired area. Initially, the markers are placed a few inches in front of the robot. A screen dump is then performed to see where the markers appear. The angle of the camera must be increased or decreased until the A and/or B markers can be seen in the screen dump along the bottom edge. Next, the markers should then be move to the outside corners of the viewable area. The C and D markers are then added to the viewing area and moved until they are in the upper corners using the same technique for positioning the other markers. The angle of the camera should not be changed while this occurs. The camera angle and marker position is recorded once the markers are in the corners of the viewing area. Marker position is with respect to the front center of the robot.

In the next step, the A and B markers should then be moved to the C and D positions or slightly below. The process presented above should be repeated. This will provide information on obtaining overlapping obstacle data.

The entire process is repeated iteratively until the distance of the extreme marker (C or D) is at least half the distance of the room being used for testing. In this case the room size is approximately 12 x 12 feet.

7.3 Obstacle Detection By Color

This experiment is designed to determine how well the CMUcam can recognize different colors within a grid using the color information returned by the camera. Two

kinds of grids are used. One with four rows and one with four rows of four squares. This partitioning may be used for obstacle detection while the robot is moving and to pinpoint the location of an object. The rectangles and squares are with respect to the logical image coordinates of the camera.

A Java application was written to position the camera, obtain the color information and analyze these results. The Java application can run the calibration routine, determine the color of the floor and capture row and square grid image data. The color of the floor, in terms of hue, is determined by analyzing the two squares in the grid directly in front of the robot.

The camera was set to a 35-degree tilt angle and facing directly forward. The system is then calibrated using the 16% gray card.

The experiment starts by obtaining the grid data for the floor alone. Five different, five-inch square colored cards were placed one at a time in the front, middle, back, and sides of the image area. Grid data is obtained for each card placement. The grid data for each test is compared to the floor hue and an object/card is identified by the difference in the hue values recorded. Initial tests assume a difference of 20 is sufficient to indicate an object.

7.4 Pivot Angle And Camera Coverage

The object of this experiment is to show that the center view of the CMUcam mounted on robot facing directly forward is not the center of the view area of the camera. This experiment will also determine what angle, if any, the camera should be rotated so the center of its view matches the area directly in front of the robot. This will allow the system to detect obstacles in front of the robot.

A five-inch, blue colored, square card was placed six-inches away from the front of the robot. The camera was set to 35-degrees. The camera was calibrated and a floor hue is obtained with nothing in front of the robot within the viewing area of the camera.

The current pivot angle is recorded with respect to the robot. Right is 0º, forward is 90º and left is 180º. An obstacle report is obtained. The camera is then rotated 5º and the process repeated until the obstacle report shows the obstacle to be in the center of the square grid data. The angle for this report is saved. The process is repeated testing angles close to this angle to determine the granularity of the obstacle report. The process is repeated again to determine when the camera view no longer detects the card.

7.5 Static Obstacle Detection

The Obstacle Detection By Color experiment used colored cards that were placed flat on the floor. This was easy to do but does not always reflect the kinds of obstacles the robot will encounter. This experiment is performed in the same fashion except that the square cards are replaced by objects that a measurable vertical component. In particular, the robots were made skirts of the same color as the cards in the prior tests. The Obstacle Detection By Color experiment was repeated using a robot sporting a colored. The skirt covered the main frame and the sides of the wheels but it did not cover the top area where the camera is located or the area where the battery holders are located. The experiment should also be repeated with colored, six inch cubes that will be used in navigation tests.

7.6 Obstacle Detection Speed

Obstacle detection speed limits the speed that the robot can navigate around obstacles. This experiment determines how much time it takes to perform these actions. This experiment also determines how many obstacle reports can be obtained in one second using the row and square program.

The Java test application will perform as obtain as many obstacle detection reports as possible within a ten second period and report the results. The Javelin Stamp only supports integer values so a long period of time is required to report fractional results.

7.7 Obstacle Detection While Moving

This experiment has the robot moving within a room while avoiding obstacles. The robot will use the CMUcam to identify objects around it. The results are more difficult to quantify because they will be based on various refinements of a navigation program but the end result is to determine whether the system can navigate through the room without colliding with an obstacle. Object detection will not be performed while the robot is moving. The robot will try to move forward from its initial position, pivoting left or right to avoid obstacles. Audible status information is supplied while the robot is operating.

The robot starts in a standing position. There must not be any obstacle within six inches directly in front of the robot. The robot takes a picture of the area making sure there is no obstacle detected.

The exploration program then examines the area in front using the CMUcam at a 15 degree angle, the minimum angle chosen for close range detection. If no obstacle is

detected the robot proceeds forward within the range of the detection area. This process is repeated until an obstacle is detected.

When an obstacle is detected the robot reads off the coordinates where the objects are detected and the color of each obstacle. Experimental data will record the actual obstacle color and location versus the reported information. After this is completed the robot pivots 90 degrees to the left or right depending upon prior pivots. The program will attempt to turn toward the direction it originally faced. These steps continue making the robot move in a zigzag form throughout the area.

7.8 Target Acquisition

In this experiment the robot searches for a colored target using the CMUcam's Track Color function. This obstacle identification is designed to locate a selected obstacle at a distance using one of the greater pivot angles than that used in the prior experiment.

The Track Color function takes the minimum and the maximum RGB values. The target color should be within this range. The function returns the view coordinates of a bounding rectangle when a matching object is found. It will also returns a confidence factor and the number of pixels within the rectangle. A high confidence factor means the color of the area closely matches the color range. The low end is 0 to 8 which means that there is a poor chance of the area matches the target color. A value higher than 50 indicates a good match based on the documentation. The experiment will determine what confidence factor provides adequate results along with the color range necessary to recognize the target.

This program uses the navigation and obstacle support from the prior experiment. It will attempt to locate a target and move towards it. The Track Color function is use for target location but not for navigation and obstacle detection.

The program starts by determining the floor color. Audible output is used to indicate when a piece of colored paper that matches the color of the target is to be placed on the floor in front camera. The robot will start searching after the paper is removed.

The robot starts by looking up and around for the target using the CMUcam. The camera pivot angle is change for long distance area coverage. The camera is then moved left and right so that three areas will be in view. The robot will pivot in the direction of a target if one is detected, otherwise, it will move forward. The target will be placed at least one yard away and in front of where the robot starts. Obstacles will be distributed throughout the area as well. There will be sufficient room between obstacles for the robot to pass through a collection of obstacles. Audible output, such as “target detected”, will be used to provide result feedback.

8. Results

The results for the following experiments are covered in this chapter.

1. Profile Gray and Robot Colors
2. Profile View To Floor Mapping
3. Obstacle Detection By Color
4. Pivot Angle And Camera Coverage
5. Static Obstacle Detection
6. Obstacle Detection Speed
7. Obstacle Detection While Moving

The experiments were performed a number of times to determine that the results were repeatable. The results presented here are for a specific set of experiments that were performed. Results from tests that were not reported were within 10% of those presented.

8.1 Profile Gray and Robot Colors

The following three tables show the results obtained from the EBot using the procedure describe in the prior chapter. There are six numbers returned by the CMUcam GM command. These are the mean red (R), green (G), and blue (B) mean values for the area examined, in this case the entire viewing area, followed by the respective deviation for each color (dR, dG and dB). A large deviation, over 30, indicates a wide variance in the color. A small deviation indicates a relatively consistent color throughout the area. This color information format is used throughout the results of all the experiments.

All values are 8-bits. The possible RGB values tend to be with the range of 20 through 250. The possible deviation values range from 0 to 120.

Trial 1	R	G	B	dR	dG	dB
Floor	145	141	109	25	22	13
Orange	240	58	21	1	15	9
Yellow	238	240	130	5	1	15
Green	99	240	190	9	1	15
Pink	240	34	84	1	14	19
White	224	240	239	24	1	14
Blue	112	217	236	26	19	9
Lime	216	249	156	19	1	18

Table 1: Profile Trial 1

Trial 2	R	G	B	dR	dG	dB
Floor	148	137	105	29	19	10
Orange	240	58	21	1	16	11
Yellow	238	240	126	7	5	19
Green	100	242	185	9	1	18
Pink	240	41	87	1	15	22
White	243	246	233	7	3	17
Blue	117	211	237	25	19	6
Lime	216	256	149	22	2	16

Table 2: Profile Trial 2

Trial 3	R	G	B	dR	dG	dB
Floor	147	143	108	25	20	14
Orange	238	60	21	1	17	10
Yellow	238	240	131	6	3	15
Green	97	238	186	9	1	17
Pink	237	36	82	3	16	21
White	240	240	236	5	1	14
Blue	115	209	237	27	20	8
Lime	215	260	157	21	3	17

Table 3: Profile Trial 3

The results were relatively consistent in each trail so an average was computed as shown in the following table. Additional trials were performed and compared to the average. Although this type of averaging is not mathematically proper, it was thought that it might provide an easy way to identify a particular color in future experiments.

Average	R	G	B	dR	dG	dB
Floor	147	140	107	26	20	12
Orange	239	59	21	1	16	10
Yellow	238	240	129	6	3	16
Green	99	240	187	9	1	17
Pink	239	37	84	2	15	21
White	236	242	236	12	2	15
Blue	115	212	237	26	19	8
Lime	216	255	154	21	2	17

Table 4: Profile Averages

Additional mathematical manipulation of the results is covered in the Discussion and Conclusions chapter.

8.2 Profile View To Floor Mapping

The results include the color information, the size and position of the obstacle (when placed) and the difference between the color information when there is an obstacle and when there is no obstacle. The camera was placed at a certain angle and then the markers would be placed at the four corners of the picture that was sent back to the computer. The angles were 15, 35, and 70 degrees.

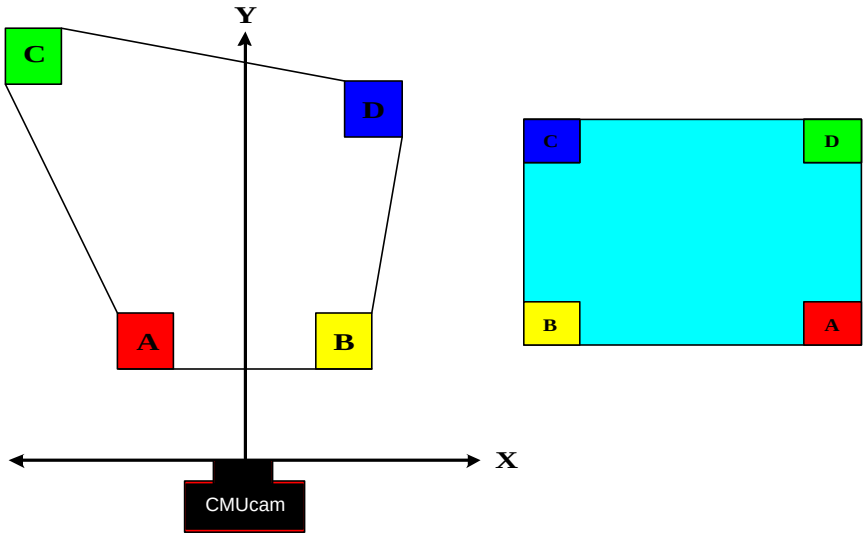

Figure 5: CMUcam Image Distortion

The coordinates of the markers shown in the figure above are enumerated in the following tables for each angle.

	X	Y
A	-8	2
B	2	3
C	-11	11
D	2	11

Table 5: Floor Profile 15 Results

	X	Y
A	-9	8
B	1	9
C	-18	26
D	6	25

Table 6: Floor Profile 35 Results

	X	Y
A	-11	13
B	5	17
C	-93	151
D	11	145

Table 7: Floor Profile 70 Results

The three angles provide overlapping image areas. The views from these angles make them suitable for use in mapping with checks at the boundary areas where overlap occurs.

8.3 Obstacle Detection By Color

The camera was placed at a 15-degree angle. The camera was calibrated and for each test there was a different colored sheet of cardboard. The sheets were placed five inches away from the center of the robot. The yellow markings in the charts show where the values are different from the hue compared to the floor readings by more than 20, the row values in the table are the hue for the row followed by the hue for each of four squares across the row.

Floor				
46	44	44	47	51
46	43	43	49	49
43	41	42	43	44
44	42	43	43	45

Table 8: Floor readings

Green				
48	50	45	46	50
84	122<	130<	52	48
72	94<	113<	49	44
45	44	44	44	45

Table 9: Green Obstacle 5 inches from center

Blue				
50	48	53	49	51
102	95<	171<	137<	49
72	58	141<	98<	44
44	43	43	44	45

Table 10: Blue Obstacle 5 inches from center

Pink				
35	29	26	37	50
20	14<	0<	21<	48
28	28	14<	25	44
44	44	44	43	45

Table 11: Pink Obstacle 5 inches from center

Orange				
42	34	38	46	50
22	5<	1<	40	48
27	20<	10<	36	44
44	44	44	43	45

Table 12: Orange Obstacle 5 inches from center

White				
46	43	44	47	51
49	49	46	50	49
52	60	60	52	45
51	52	57	52	45

Table 13: White Obstacle 5 inches from center

Brown Box				
46	42	42	46	51
41	47	40	35	42
46	48	47	47	43
47	47	47	46	45

Table 14: Brown Box Obstacle 5 inches from center

8.4 Pivot Angle And Camera Coverage

The following tables show the results from the Pivot Angle and Camera Coverage test. The angle is noted in the upper left corner. The row values in the table are the hue for the row followed by the hue for each of four squares across the row. Values that are different from the floor hue by more than 20 are noted by the angle bracket (<) and the cell color of yellow.

90 degrees				
54	42	55	60	59
53	42	50	58	58
59	42	41	64	91<
48	42	39	48	70

85 degrees				
54	43	53	60	59
52	43	47	59	59
56	43	40	50	101<
48	43	40	41	75

80 degrees				
54	44	50	60	60
52	45	45	58	59
51	46	42	47	74
44	45	41	40	54

75 degrees				
53	43	51	60	60
52	47	46	55	59
48	47	44	44	54
42	45	44	39	41

95 degrees				
55	43	57	60	56
53	41	51	59	58
59	40	43	90<	67
47	40	39	61	49

100 degrees				
56	44	59	60	58
54	42	55	59	56
58	39	57	107<	45
45	38	44	64	38

105 degrees				
56	48	59	60	57
54	43	56	58	56
60	40	95<	76	44
43	38	49	45	39

110 degrees				
56	50	59	59	57
55	45	61	57	55
63	45	139<	47	44
41	39	47	38	44

115 degrees				
55	51	59	58	51
56	57	59	54	52
59	76	90<	43	50
41	40	39	40	46

120 degrees				
55	52	58	56	51
61	87<	54	52	56
57	100<	44	46	55
42	39	37	42	42

125 degrees				
53	52	57	57	53
61	96<	47	51	58
53	76	39	49	57
41	38	38	46	41

130 degrees				
55	51	56	55	57
55	70	44	53	60
48	44	40	51	55
40	36	42	45	39

135 degrees				
53	47	52	50	57
50	44	46	56	60
48	37	44	55	54
40	38	42	40	40

Table 15: Obstacle Detection at given degrees

8.5 Static Obstacle Detection

The following are representative results from the tests that employed a robot with a colored paper skirt. The results shown were using a robot with a blue color. The row values are the same as the results of prior experiments including the color and angle bracket (<) annotation of the obstacle based on hue differences greater than 20 from the floor values.

Floor				
47	38	40	55	58
45	42	43	44	50
40	41	41	43	34
38	39	40	37	36

Center				
47	37	39	50	58
55	40	150<	126<	48
58	48	170<	152<	36
75	85<	183<	170<	39

Right Side				
39	38	40	45	35
45	42	42	39	145<
43	41	39	41	161<
43	40	40	40	176<

Left Side				
52	66<	41	55	58
47	168<	41	42	49
39	177<	40	40	34
38	92<	39	37	36

Table 16: Static Obstacle Detection Of A Blue Robot

8.6 Obstacle Detection Speed

The robot can detect an obstacle in a row or square at the rate of 2.2 checks per second. This translates to 1.8 seconds per 4-row obstacle report and 7.2 seconds per 16 square obstacle report. Unfortunately, the reliability of the 4-row report was poor for obstacles were partially in the row or for obstacles that are small. The 16-square report is necessary to properly identify obstacles reliably. This did lead to a slow overall movement of the robot.

8.7 Obstacle Detection While Moving

The top speed of the robot is 10 inches per second. The first navigation method has the robot obtaining its obstacle information from a standing position. There was no problem detecting obstacles in this fashion and the process took 7.2 seconds using a 16 square grid (2.2 seconds/square). Each square corresponded to at least a 5 inch square area. For ease of implementation, a square was assumed to be 5 inches and the length of the grid was 20 inches. At top speed, the robot covered the distance in 2 seconds. The overall, maximum forward speed was 9.2 seconds for 20 inches or 2.2 inches/second.

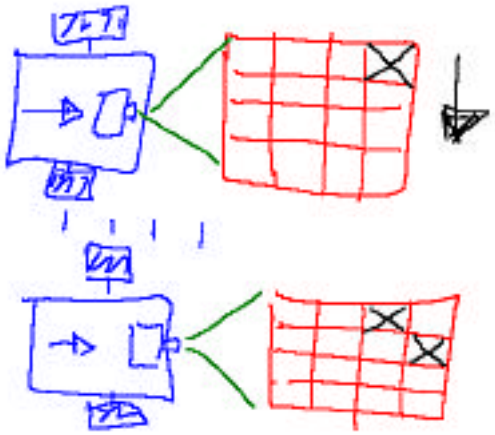

Figure 6: Obstacle Detection While Moving

The figure shows the effects of doing obstacle detection while the robot is moving. The robot begins its scan and looks for an obstacle in a square on the left side of the grid. By the time it can scan the next square the robot has moved forward. If the next square is in the same logical row as the first then the actual position will be farther away. The alternative was to scan diagonally across the logical grid to approximate scanning across the actual row area in front of the robot. In either case, the time it took to scan a row was 1.8 seconds. This allowed forward movement at a maximum rate of 2.5 inches/second.

When the experiment was ran the robot would send back information that said an unknown obstacle was detected. This meant that the value was outside the range of the known solid colors, which indicated the object took up only part of a pixel.

8.8 Target Acquisition

Initially the robot was usually unsuccessful in detecting the assigned target with a pivot angle of 35 degrees and an RGB delta of 5. Also, the robot can not detect the target

unless it is a minimum distance in front of it. This distance was about one foot. These results were unacceptable for general use so the test parameters were changed.

The robot was more successful wen the pivot angle was increased to 70 degrees and the RGB delta was set to 20. Targets that were one foot to six feet away were detectable but not every time. Distance did not affect the results but lighting did.

The pivot angle was maintained but RGB delta was increased to 60. This improved repeatability significantly although accuracy was not 100%. It was over 80% and the accuracy increased as the target was moved closer to the robot.

9. Analysis

The CMUcam is a low cost camera that can provide sufficient information to do limited visual scene analysis and identify objects by color. The system developed is not as powerful or precise as the one described in *Appearance-Based Obstacle Detection with Monocular Color Vision* (Nourbaksh and Ulrich, 2000) but this system is suitable for use on smaller robots such as the EBot. The translation of RGB (red, green, blue) color information to hue, saturation and intensity (HSI) information helps to overcome the problem of varying illumination (Rowe, 2001b).

9.1 Analysis of Experimental Results

The *Profile Gray and Robot Colors* experiment provided insight into the quality of information that could be provided by the CMUcam. The results were relatively consistent under the same lighting conditions but changing lighting conditions resulted in color results that were comparable for colors that were darker or lighter.

One thought was to compute the range of the averages because comparing mean values and deviations to get a confidence factor tended to be difficult. The resulting table is based on the test averages.

Range	R low	R high	G low	G high	B low	B high
Floor	120	173	120	161	95	120
Orange	238	240	43	75	11	31
Yellow	232	244	237	243	113	145
Green	90	108	239	241	170	204
Pink	237	241	22	52	64	105
White	224	248	240	244	221	251
Blue	89	141	193	232	229	244

Table 17: Profile High-Low Results

Unfortunately, these ranges tend to vary when lighting conditions vary. This includes affects from shadows and different viewing angles. One way to get around this problem is to translate the RGB information so the intensity of the color is not a factor.

The first step is to convert the RGB information of HSI (hue, saturation, intensity). HSI is often used on television sets to tune the color of the image because the television signal is transmitted in HSI format. Once in HSI format, the intensity component can be ignored thereby ignoring the effects of lighting and shadows.

The next table shows the HSI values for average RGB results presented earlier.

HSI	Hue	Sat	Intensity
Floor	220	39	127
Orange	248	221	130
Yellow	212	198	184
Green	143	208	169
Pink	10	218	138
White	0	0	239
Blue	119	194	176
Lime	196	250	204

Table 18: HSI of Average RGB Values

This table shows the HSI value for the low range RGB values presented earlier.

HSI Low	Hue	Sat	Intensity
Floor	213	30	108
Orange	249	232	125
Yellow	211	195	175
Green	147	208	164
Pink	8	217	130
White	176	97	231
Blue	117	184	159
Lime	191	242	195

Table 19: HSI for Minimum RGB Values

Note the similarity in the hue values with the exception of the white color. The hue and saturation values are 0 while the intensity is very high. The intensity is high in the HSI Low table above but the hue and saturation are not zero.

This is actually reasonable given the way colors work in an HSI system. A very low intensity is a dark color with black being at the extreme end. The opposite end of the spectrum is white. In these extremes, the intensity value dominates. The 0 value is actually due to the way the HSI values are calculated.

The following table shows the HSI values for the maximum RGB values obtained in the experiment.

HSI High	Hue	Sat	Intensity
Floor	222	62	146
Orange	246	222	136
Yellow	213	205	195
Green	140	208	174
Pink	12	219	146
White	0	0	247
Blue	123	208	192
Lime	203	0	214

Table 20: HSI for Maximum RGB Values

Note the saturation value for the Lime color. It turns out that it has a high intesity value (above 200) as well. This indicates that the Lime color is very bright. This turned out to be true when a screen dump from the camera was viewed on the PC. The image was very bright and close to white with a slight green tinge. This was close to the camera's limits and the numeric results reflect this.

The use of hue information worked well with the hardwood floor. No additional filtering or processing was required since the mean color results for an area already averaged the mix of colors on the floor. Similar results are expected for the other colors tested assuming different textures or mix of similar colors was employed. The averaging would not provide comparable results if an object's surface had a wide range of colors.

The following Java methods were used to calculate the hue, saturation and intensity.

```
/**
   * Compute hue
   *
   * @param r red
   * @param g green
   * @param b blue
   *
   * @returns hue
   */
  static public int hue ( int r, int g, int b ) {
    int h ;
    int min = Math.min ( r, Math.min ( g, b ));
    int max = Math.max ( r, Math.max ( g, b ));

    int intensity = (min+max)/2;
    if ((min == max) || (intensity < 20) || (intensity > 255)) {
      return 0;
    } else {
      if (r == max)
        /* color is between yellow and magenta */
        h = (60 *(g -b))/(max-min);
      else if (g == max)
        /* color is between cyan and yellow */
        h = 120 + (60*(b-r))/(max-min);
      else
        /* color is between magenta and cyan */
        h = 240 + (60*(r-g))/(max-min);

      // Normalize to 0-359
      while (h < 0) {
        h += 360;
      }
      while ( h > 360 ) {
        h -= 360;
      }

      return h;
    }
  }
```

Code 1: Java Hue Method

The hue is often referred to as a color circle and the hue is actually calculated in terms of 360 degrees. This must be taken into account when comparing hue values because the difference between a value of 350 and 10 is actually 20.

```
/**
   * Compute saturation
   *
   * @param r red
   * @param g green
   * @param b blue
   *
   * @returns saturation
   */
  static public int saturation ( int r, int g, int b ) {
    int h ;
    int min = Math.min ( r, Math.min ( g, b ));
    int max = Math.max ( r, Math.max ( g, b ));

    int intensity = (min+max)/2;
    if ((min == max) || (intensity < 20) || (intensity > 255)) {
      return 0;
    } else {
       if (i <= 127) {
         return ( 255 * (max – min) / (max + min) ; // dark color
       } else {
         return (255 * (max - min) / (512 - max - min)) ; //
bright color
       }
    }
  }
```

Code 2: Java Saturation Method

The saturation is a percentage although this method actually converts the result into an 8-bit value. This turns out to be easier to manipulate especially when converting an array of 8-bit RGB values to HSI values.

```
/**
   * Compute intensity
   *
   * @param r red
   * @param g green
   * @param b blue
   *
   * @returns intensity
   */
  static public int intensity ( int r, int g, int b ) {
    int min = Math.min ( r, Math.min ( g, b ));
    int max = Math.max ( r, Math.max ( g, b ));

    return (min+max)/2;
}
```

Code 3: Java Intensity Method

The intensity is simply the average of the minimum and maximum RGB values. The method is not used in the hue and saturation methods because it would result in the

redundant calculation of min and max. In fact, if an array of RGB values must be converted to HSI values then another method that performed all three conversions would be more efficient.

This experiment did not validate the use of HSI for detecting obstacles but it did show promise. In fact, the hue value alone appeared to be a good way to differentiate different color objects. Subsequent experiments bear this out.

A zero value will be returned for both the hue and saturation when the intensity is very low or high. The intensity comes into play in this case making it the dominant factor in determining the difference between colors.

The *Profile View To Floor Mapping* test confirmed the problem of translation between the actual area on the floor and the image used by the CMUcam. Adjusting for this translation is not a difficult process as the distortion is still linear. Use of a fixed grid will allow the translation to be performed once and applied to the results obtained from the CMUcam.

The *Obstacle Detection By Color* confirmed the use of hue information for easy obstacle identification using the variety of colors selected for the tests. A single comparison is needed to determine that a colored object is different from the floor or other objects.

The test highlights the need for a fine granularity for obstacle detection. The change in a row result was small if only one of the squares in the row has a major deviation from the floor hue. This indicates that using the row information is insufficient to identify an obstacle and the more precise square grid is needed to identify obstacles.

Being unable to use the raw results was disappointing because it meant that multiple squares must be evaluated slowing down performance by a factor of 4. This problem becomes more acute if the number of squares in the grid are increased.

On the other hand, the 16 square grid proved sufficient to identify obstacles that are within the size range for the test environment. This method can be used to easily identify another robot or an obstacle that is even half that size.

Likewise, finer positional information can be obtained by using a finer grid. The process may be slower but the added time may be offset by the higher precision.

The results from the *Pivot Angle and Camera Coverage* tests prove that the center of the robot is not the center of the camera. The yellow boxes show where the obstacle was detected at the given degree. The results from 105° and 110° show the detected object near the center.

The *Static Obstacle Detection* test showed that simply moving to 3D objects did not affect the accuracy of identifying objects. In fact, a 3D object appear larger because of the object's height. This did highlight the problem of limiting obstacle position information to the front of the object. No attempt has been made at this time to determine the height of an object.

The *Obstacle Detection Speed* test proved to be the most disappointing. It showed how slow the system performed compared to the speed of the robot. It was hoped that the camera system would be able to provide obstacle information on the fly with the robot running at full speed. The test did prove that obstacle information could be obtained within a couple of seconds though.

The *Obstacle Detection While Moving* test showed that the obstacle detection system was sufficient for navigation even the robot could not move continuously at high speed. The visual information utilize in the test was sufficient for mapping purposes although this was not performed in this test. Another thing that effected some results were the shadows of the objects onto the floor. Although the shadows were on the floor they camera saw a difference in color. The camera calculates the mean value using RGB and not hue or saturation intensity. It is better using the hue and saturation intensity rather then the RGB so the mean values wouldn't have to be converted.

Finally, the *Target Acquisition* experiment showed that lighting had a major impact on this experiment. Although the colored paper is placed in front of the robot at beginning of the test, the RGB-based Track Color function of the CMUcam did not allow the robot to readily locate the target until the range of RGB values was increased significantly. The track color system worked pretty well except for when the range of the values were extremely small due. This is due to the fact that the lighting was different with respect to the target that would often cast a shadow on itself thereby changing the detected RGB values. When the program was adjust it was finally successful at being able to detect the target and to turn towards the target's location to get closer to the target. Overall the robot and the program was able to detect the target.

10. Conclusion

The basic visual navigation and object detection system based on the CMUcam was a success. It was able to identify objects by color and provide sufficient information to navigate around them. The system operates more slowly than desired but improvements may be possible.

The use of HSI information was superior to the use of RGB information. Unfortunately, the CMUcam only provides RGB information requiring additional communication and processing of the data.

The system does not provide sufficient information for object identification by scene analysis other than by color and it does not appear that this will be possible based on the capabilities of the CMUcam. Using a different camera may provide the resolution necessary for this type of obstacle recognition but at a higher cost for the hardware and higher processing requirements.

The system can be used to identify different shades of colors that are close to each other but additional experiments must be done to determine how close the differences can be. Likewise, additional tests need to be done using more varied lighting conditions to see how well the hue information is maintained.

11. Future Work

I hope to continue with my robotic experiments over the next year. I want to take multiple robots I have constructed and have them work together so each robot searches their own areas. Communication between robots will be done using the wireless link.

Also I would like to reprogram the CMUcam camera so it operates using HSI values instead of RGB. This will reduce the amount of calculations and making it easier to work with the results. It may also allow real time navigation instead of the start-stop mode currently used.

12. Bibliography

Abbot, B., Torrie, M. W., and Veeramachaneni, S. (2000). *Laser-Based Obstacle Detection and Avoidance System,* CSOIS, Utah State University.

Chen, Xing, and Davis, James, (2001). *A Laser Range Scanner Designed For Minimum Calibration Complexity*, Computer Graphics Lab, Stanford University.

Diehl, C. P., Hampshire II, J. B., Khosla, P. K., and Sapthrarishi, M. (2000). *Collaborative Surveillance Using Both Fixed and Mobile Unattended Ground Sensor Platforms*, Department of Electrical and Computing Engineering and ICES, Carnegie Mellon University.

Dolan, J. M., and Ollennu-Trebi, A. (2000). *An Autonomous Ground Vehicle for Distributed Surveillance: CyberScout,* Institute for Complex Engineered Systems (ICES), Carnegie Mellon University.

Druin, A. and Hendler, J. (2000). *Robots for kids Exploring New Technologies for Learning,* Morgan Kaufmann, San Francisco, CA.

Keller, P., and Singh, S. (2000). *Obstacle Detection For High Speed Autonomous Navigation,* Field Robotics Center, Carnegie Mellon University.

Hancock, J. A. (Jan. 1999). *Laser Intensity-Based Obstacle Detection and Tracking*, The Robotics Institute, Carnegie Mellon University.

Hirschmüller, H. (2001). Improvements in Real-Time Correlation-Based Stereo Vision, *Proceedings of IEEE Workshop on Stero and Multi-Baseline Vision*, pp. 141-148.

Nourbaksh, I. and Ulrich, I. (2000). *Appearance-Based Obstacle Detection with Monocular Color Vision*, The Robotics Institute, Carnegie Mellon University.

Rowe, A., Rosenberg, C. and Nourbaksh, I. (2001). *A Simple Low Cost Color Vision System,* The Robotics Institute, Carnegie Mellon University.

Rowe, A. (2001). *CMUcam Vision Board User Manual*, The Robotics Institute, Carnegie Mellon University.

Rowe, A. (2001). *Illumination and its effect on CMUcam,* The Robotics Institute, Carnegie Mellon University.

Trebi-Ollennu, A. and Dolan, J. (1999), *An Autonomous Ground Vehicle for Distributed Surveillance: CyberScout*, Carnegie Mellon University.

Wong, L. (2002). *Where Am I? Minimizing positional error while navigating and mapping using a cooperative robotic system,* Villa Victoria Academy.

13. Acknowledgements

I would like to thank Parallax Inc. who have sponsored and supported my projects for the last two years. I would like to thank my friends and teachers who also supported me throughout the months of work I put into these projects.

I would also like to thank each of my family members personally. To my sister, Jennifer, and brother, Robert, who were the ones who got me interested in robotics and for being there for me every step of the way. To my mother, Ann, who encouraged me to keep going and to reach that extra mile.

Finally to my father, Bill, who was there everyday when I argued and complained and told me to never give up.

Thank you to all of you. Without you this project would not be possible.