Implementation of an Embedded System For Vehicle Avoidance For a BMW X5 System

Jonathan Eng
20263557

School of Electrical, Electronic and Computer Engineering

Supervisor: Professor Thomas Bräunl
School of Electrical & Electronic Engineering, University of Western Australia

Final Year Project Thesis
School of Electrical, Electronic and Computer Engineering
University of Western Australia

Submitted: November 4th 2011

Contents

Abstract

In this project I was successful in fully implementing stage one of the vehicle detection algorithm onto the Eyebot M6 embedded controller designed by the Automation Computing and Energy Research Group (ACE Group). I had to construct an serial RS232 – to – TTL converter so that the Eyebot M6 would be able to send the signals to the Arduino microcontroller that controls the braking and steering motors on the BMW X5 vehicle. I was also able to design a simple user interface which gave the driver the ability to choose between two modes of resolution and the option of activating the steering avoidance system and deciding which direction to turn.

The whole programming aspect of the Eyebot M6 was done using Microsoft Visual Studio 2008, and the entire code revolved around the symmetry based detection algorithm that stage one of the vehicle detection algorithm is centred on. For the signals to be sent out to the Arduino controller, I had to work with a mechatronics student who designed and implemented the braking and steering systems on the BMW X5. We had to devise a solution to send the command signals to the Arduino board from the Eyebot M6, and had the capability of handling glitches in the commands as well as to be noise resistant when sending out commands in the channel. The Eyebot M6 could not send the signals out using the USB port as it was faulty but was able to send the signals via the serial port, however the Arduino board had no serial RS232 ports but had TTL ports. To overcome this, I had to construct an RS232 – to – TTL converter to convert the signals so that the Arduino board could processes the commands. The braking protocols I used when a vehicle/object was deemed to be too close to the vehicle without any other inputs other than the camera, was that if said vehicle/object's base was below a set row on the image, the vehicle would start braking. The same concept is also followed by the steering protocols.

Without stage two of the vehicle detection algorithm, the system can only assume that the detected object is a vehicle, as stage two checks the detections from stage one with a database of known vehicles. As such the system treats all detected objects within the region of interest to be a valid vehicle.

Acknowledgement

I would like to thank my family for their unending support throughout the project. I would also like to thank my friends for keeping me sane during the final weeks of the project and especially to the guys in my office Teoh and Jithun for teaching me new things and discussing about anything and everything during our coffee and lunch breaks.

I would especially thank Professor Thomas Bräunl for giving me this opportunity to undertake this project, as well as his constant guidance and support. A special thanks goes out to BMW for donating the X5 vehicle to UWA for research purposes, for without the vehicle it would not have been possible. Another special thanks goes to Soo Teoh, for without him and his PHD topic this project would not have been successful, and lastly I would like to thank God for the strength and wisdom to make it this far.

3 Introduction

Embedded systems are becoming more and more interconnected with our everyday lives, making us more dependent on them then we realise, from washing machines to digital watches, transportations to communications and entertainment. One industry that is fast tracking towards an autonomous future is the transport industry, with some trains in the developed world operating autonomously with minimal human intervention, using highly advanced embedded systems to operate. On the other hand the car industry is rapidly catching up with the advancements of camera vision technologies and algorithms and low cost hardware in driver assist technologies.

One of the big leaps in driver assist technologies a few years ago, was the implementation of reverse cameras in cars. These reverse camera implementations gave drivers more visual information about the rear ends and the space around them, with the more expensive cars predicting and displaying guidelines on how the car is going to be parked based on the rotational angle of the steering wheel. A more recent break through is the recent development of obstacle avoidance systems on cars to increase the safety of not only both the driver and its occupants but also for the other driver and their passengers. The Volvo car company is one of the first car companies to implement such a system, called Collision Avoidance System. Renowned for its safety standards, Volvo's system issues visual and audible warnings and provides automatic braking and steering when it senses a likely collision with another vehicle or persons **[1]**. Such a system involves the implementation of advanced high level algorithms in object detection and recognition onto sophisticated and powerful custom made embedded systems.

The reason for the research into driver assist technologies in cars is to lower the probability of accidents caused by preventable variables on the driver, such as fatigue, inattentiveness and impairment. It is estimated that there are around 1.2 million deaths and 20-50 million injuries from road related accidents around the world each year caused by factors such as driver fatigue, inattentiveness and impairment **[2]**. Recently there has been an increase in the number of research going into driver assist technologies that minimizes accidents, such as force feedback in the steering wheel when the vehicle crosses lanes with no indicators which is processed by lane recognition and tracking algorithms **[3]**, displaying the speed limit and warning the driver if they are speeding on a dynamic display which doesn't distract the driver

[4], lane tracking and lane keeping algorithms using camera based image processing algorithms which activates when it detects inattentive and impaired drivers, keeping the car from going off the road and crossing lanes by controlling the steering wheels and brakes **[5]**. Currently in the BMW driver assistance systems group at UWA, work is being done into the development of a symmetry-based vehicle detection system which uses a single camera to capture live video feed to be processed.

This project will deal specifically in the implementation of a symmetry based detection algorithm that uses a camera input of the scene in front of the car to determine if there is a vehicle detected in front of it and if the car is at a certain close distance to the vehicle, the system will prevent a possible collision by either braking or steering away from the vehicle. Included in the implementation is looking for possible ways to also improve the performance of the algorithm, as the algorithm was developed using a dual cored 2GHz processor and the implemented high level embedded system that the algorithm is to be ported to runs a small 400MHz micro controller.

The vehicle detection algorithm is constructed in three stages, stage one processes the image from the camera and applies an edge detection algorithm on it, it then samples each row of the image at a set interval resolution. Where it finds if there are even numbers of edge sets in that row, if there is then it is taken as a detection of interest in the image. In stage two it takes this information, the detection of interest, and compares it to a set database acquired from machine learning to determine if it is a valid detection, ie a vehicle. If at stage two the algorithm determines that it is a valid detection of a vehicle then stage three beings which tracks the vehicle using the Kalman filter approach. The vehicle detection algorithm also has an added bonus, it is able to detect and process multiple vehicles in real time **[6]**. However due to limitations of the hardware on the Eyebot M6 only stage one will be implemented, which is sufficient for detecting objects on the road regardless if it is a vehicle to be avoided, as any sizeable object on the road would generally be avoided.

Stage one of the symmetry based vehicle detection algorithm is to be ported over to a custom made embedded controller named the Eyebot M6. The M6 is the latest generation of embedded vision systems specifically designed by the ACE Group. The M6 was designed and built with real time image processing in mind, compared to its predecessors **[7]** which were built with robot control and basic image processing in mind. The M6 current hardware specs were a big step up from its predecessor, which were built with a slow 25MHz 32-bit Microcontroller, 1 MB of Ram and 512KB ROM for the system and user programs, it has four manual input buttons and a basic LCD display. Whereas the M6 utilizes a powerful combination of 400MHz ARM9 controller together with an Xilinx FPGA, which takes care of sensor/actuator pre-processing including low level image processing which would free up the main CPU for higher level tasks. Stereo cameras, a color touch LCD screen, multiple motor outputs and sensor inputs, host and slave USB, Ethernet Lan and Bluetooth communication are the main upgraded features that the M6 has over its predecessor.

The Eyebot M6 embedded controller will send out serial signals while the algorithm is running to a motor controller called, the Arduino Duemilanove, which will process the commands received from the M6 and send out the appropriate PWM signals to the motors that control the steering and braking systems. The Arduino board is a computing platform based on a simple microcontroller board, and can be used to take in inputs and outputs from switches, sensors, motors and other physical outputs **[8]**. It has a wide range of applications that can be used in building prototype electronics which interact with various outputs and interactive environments.

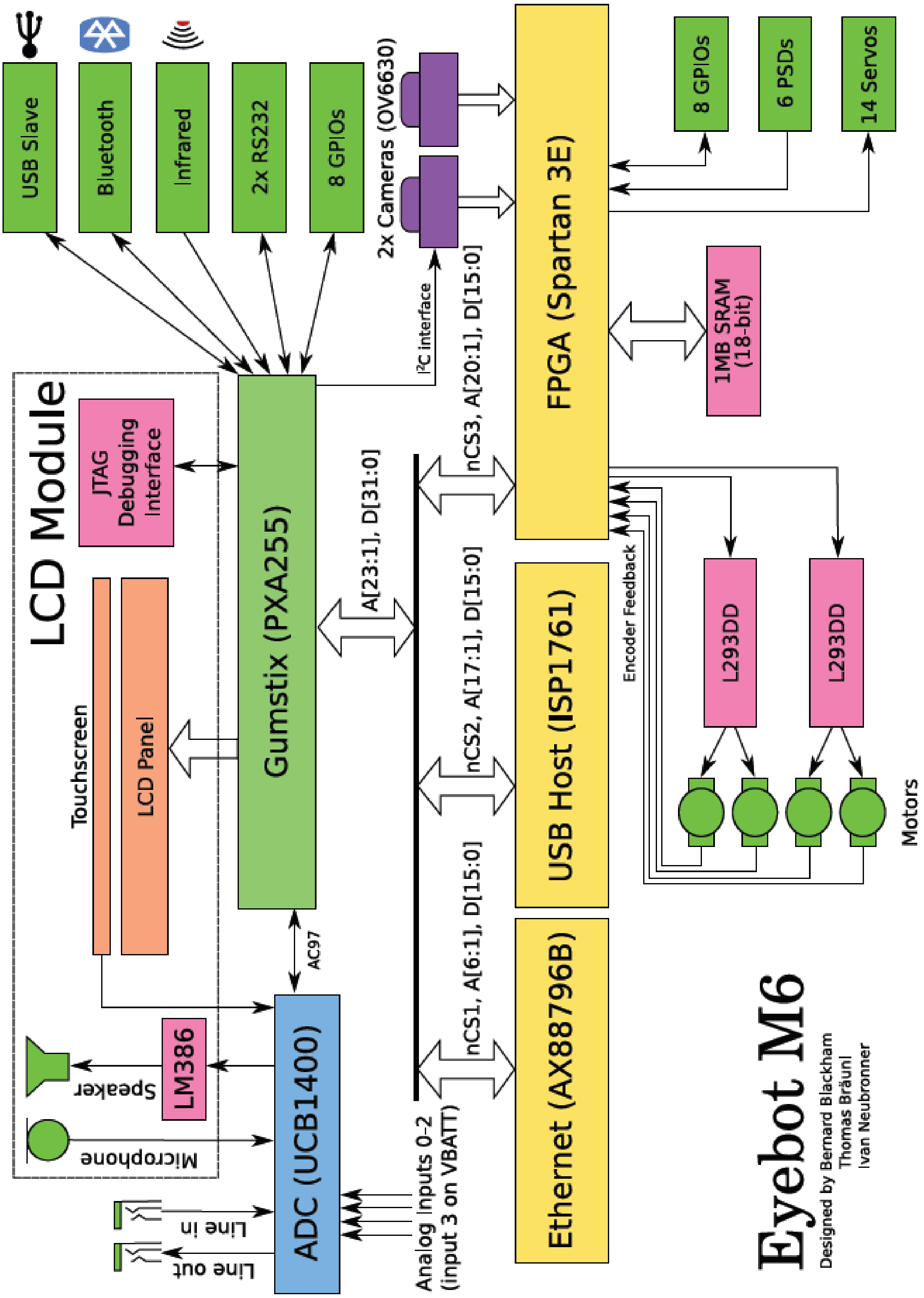


Figure 1: Layout of Eyebot M6

4 Obstacle and Vehicle Avoidance Systems

There are currently large numbers of various papers and journals on the development of obstacle and/or vehicle avoidance algorithms and many more on the implementations of these avoidance algorithms into software and physical systems. The reason for such large number of papers is because there are various ways of achieving the same goal using various techniques and algorithms. Although the majority of these papers were written not for vehicles in mind but for mobile robots, the same concepts that were applied for robotic systems can be transferable to vehicle systems.

There are two main types of obstacle detections and avoidance systems for robotic systems, indoors and outdoors. Indoor detections and avoidance systems uses simple infrared sensors that are cheap and easy to implement with accurate enough results to avoid oncoming obstacles, with indoor environments there is the added bonus of less dynamic objects in the immediate surroundings making it easier to navigate through and less dynamic lighting environments that can affect the visual systems. The addition of a camera input would allow the robotic system to detect and avoid more obstacles both static and dynamic, than just using infrared sensors **[9]**. However for this project the main focus will be on outdoor environments where the environment is dynamic compared to the static indoor environments.

There is a large base of research going into obstacle detection and avoidance systems, the majority of which are for land based vehicle systems and robotics systems and the rest are divided between aerial and underwater environments. There are some interestingly complex solutions that are implemented for land based systems, however the equipment required are on the expensive side, as these solutions require high precision data with high data density which allows the vehicle system to perform complex algorithms to react accurately and quickly to dynamic changes in the environment, such expensive equipment includes laser technologies that accurately measure distances and even graph out the surroundings in a 3-D mapping structure**[10][11]**, sonar and radar that can measure the speed and the location of objects in the surrounding area, giving accurate tracking information about objects of interest to the system**[12]**.

An interesting paper that come across during the lit review was the use of occupancy grids for obstacle avoidance in non-holonomic vehicles **[13]**, which in its implementation used a stereo camera, a scanning laser range finder and the vehicle's odometer to create the occupancy grids, which details the state of the terrain in the surrounding area of the vehicle picking up obstacles that could be avoided to prevent a collision. On another paper there was a sonar based mapping and navigation system that was developed for mobile robots operating in unknown environments **[14]**, which uses the data from sonar to build a multileveled map data of the robot's surroundings. From the sonar data, probability profiles are used to determine if the area around the robot is empty or occupied with objects and obstacles.

Although there are large collections of papers depicting the use of lasers and sonar and radar systems, there are even more on vision based obstacle detection techniques, the reasons being the lower cost in using a simple camera system and that there are many ways of approaching the same problem using different algorithms and different parameters and variables to achieve the same outcome. There are two main categories for camera vision based methods, single vision based approach and stereo vision based approach.

In single vision based systems only one camera is used, one of the key concepts used is comparing the appearance of a detected object with a database. Detection of objects are done via shape and edge recognition algorithms which are then verified to confirm that the detected objects are valid and helps reject false detections**[15][16][17]**. The accuracy of the object recognition lies in how detailed its recognition database is, as different angles of a valid object may not be recognised if the database and comparison algorithm is not sufficient.

In stereo vision based systems two cameras are used, both cameras are pointing in the same direction looking at the same scene but are at a certain distance away in parallel to each other, in order to have an epipolar line that is satisfied between the two cameras. With stereo vision, the system is able to generate a 3D data image of the scene. The stereo vision based system has the added advantage of encompassing all the functionality of single vision based systems and allows depth data in the image scene. Detections are based on stereo matching , which finds and matches the pixels from both left and right images that corresponds to the same spatial point in the scene. The depth data of the point is calculated via the disparity between the two pixels of the same spatial point **[18]**. The matching is done using either a sparse stereo matching process **[19][20]** or a dense stereo matching process**[21][22]**.

5 The Edge Detection Algorithms

Before going into discussion about the various edge algorithms that are in use, there needs to be an established definition of what edges in an image are. Edges are often found at points where there are large variations in the luminance values in the image, which mark the edges or boundaries of objects in an image. Mathematically defining edges to be step discontinuities in the image signal makes it easier to determine where edges are. By using the derivative of the signal and finding the local maxima or in the second derivative of the signal the zero crossings, the step discontinuities can be found and hence the edges in the image can be found **[23]**.

5.1 First Order differential method of Edge detection

In edge detection algorithms, edges are assumed to be where there is a discontinuity in the intensity function or a steep intensity gradient in the image. The edges are found by taking the derivative of the intensity values over the image and locate the points where the derivative is a maximum peak, however for discrete images another method is used. In discrete images the gradient can be acquired by finding the difference in grey values between adjacent pixels, which is the equivalent to convolving the discrete image with a mask. For the image function I,

The gradient is given by: $$\nabla I = \begin{bmatrix} {\partial I}/{\partial x} \\ {\partial I}/{\partial y} \end{bmatrix}$$

The magnitude by: $$\sqrt{({\partial I}/{\partial x})^2 + ({\partial I}/{\partial y})^2}$$

and the direction: $$\tan^{-1}\left(({\partial I}/{\partial y}) \Big/ ({\partial I}/{\partial x}) \right)$$

Figure 2: Equations for First order differential method

Although it is possible to use any orthogonal pairs to work out the gradient, the edge algorithms in this project uses the common *x* and *y* directional pairs, where *x* and *y* are the horizontal and vertical directions.

A major problem when using differential edge detection algorithms is the sensitivity to noise in the image. Noise produces peaks in the derivative form which can hamper the real maxima that indicates the real edges. To overcome this, a smoothing function needs to be applied to the image so that the effects of noise in the image can be reduced. A common way of smoothing the image is convolving the image with a Gaussian mask and then calculate the derivative of the new smoothed image.

5.2 Second Order Derivative

In the second derivative of the image, the maxima of the first derivative can be found at the zero-crossings. The Laplacian of I gives the capability to get both the horizontal and vertical edges, which is taken from the second derivatives of both the x and y directions. The Laplacian also has the added bonus to be both linear and rotationally symmetric.

$$\nabla^2 I = {\partial^2 I}/{\partial x^2} + {\partial^2 I}/{\partial y^2}$$

Figure 3: The Laplacian of I

With this, there are two ways to get the edges of the image, the image is first smoothed using a Gaussian mask and then have the second derivative calculated, or convolve the image with the Laplacian of the Gaussian. This is also well known as the Marr-Hildreth operator.

$$\nabla^2(G \otimes I) = \nabla^2 G \otimes I$$

Figure 4: The Marr-Hildreth operator

The Marr-Hildreth operator is widely used in computer vision techniques and algorithms for edge detections because the operator mimics the receptive fields in the eyes of some animals. The operator has the added bonus of being able to find edges in all orientations due to it being symmetric unlike the directional based derivative operators. The major advantages in using the second derivative method, is that it is more much easier to look for the zero crossings than the maxima in the first derivative, and the zero crossings of the signal always form closed contours **[24]**.

However the downside is that noise greatly affects the operator as it is a second derivative operator, which is sensitive to the disruption of noise. The other downside is the formation of always closed contours is not realistic and that the operator will generate responses to areas with false edges.

5.3 Thresholding Algorithm

The Thresholding algorithm used in this project, takes in an 8-bit grey scaled image and then compares each pixel with a set threshold level, where the pixel value is the grey scale value at that pixel. If the pixel value is found to be above that threshold, then that individual pixel will be set to a chosen value, however pixel values that are below the set threshold level will be set to 0. The resulting image produced from this algorithm is a binary image, or more accurately a 2 grey level image. The advantage to this algorithm is that it is fast and simple for the system to process, however the downside is that the algorithm is very sensitive to differences in light intensities which is detrimental to the clarity of the image.

$$FinalImage(x,y) = \begin{cases} Chosen\,Value, & SourceImage(x,y) > Threshold \\ 0, & Otherwise \end{cases}$$

Figure 5: Threshold equation

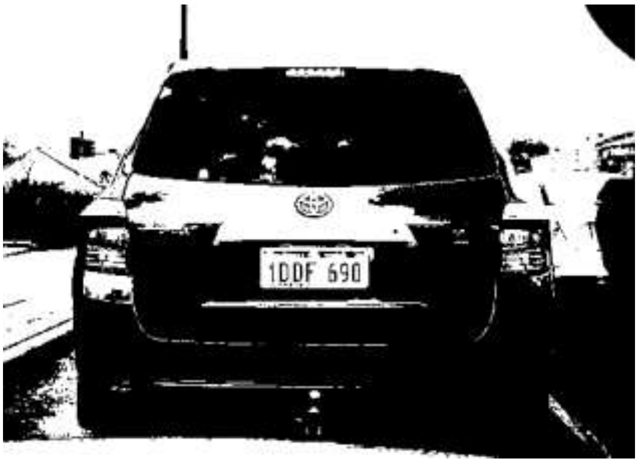


Figure 5a: Threshold image

5.4 Laplace Algorithm

The Laplace algorithm is a second order derivative that detects the zero crossings from the second order difference of the image. See the Second Order Derivative section.

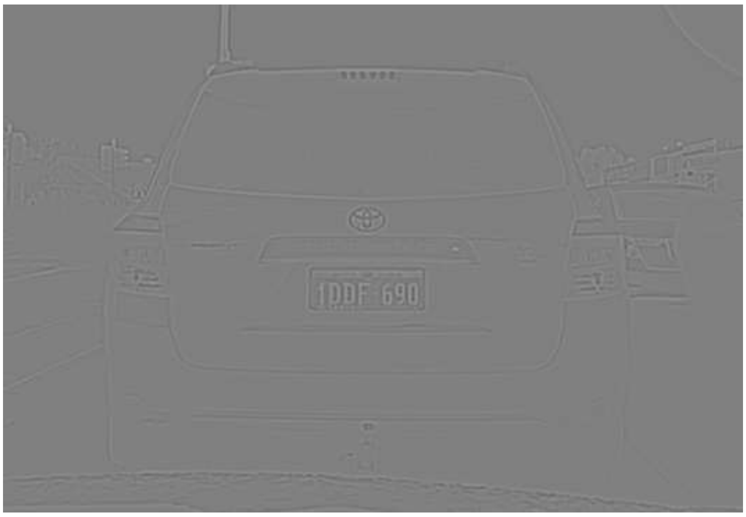


Figure 6: The Laplace Edge Image

5.5 Sobel Algorithm

The Sobel algorithm uses the Sobel operator which is a first order discrete differential operator that calculates the gradients of the intensity function and its directions in the horizontal and vertical directions. The Robert's Cross Operator uses the diagonal directions to calculate the gradient vectors of the image intensity function, and is regarded to be the simplest gradient operator.

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} and \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Figure 7a: Robert's Cross Operator

The Prewitt uses the vertical and horizontal directions to calculate the gradient of the image intensity function, using a 3x3 convolution mask.

$$\begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

Figure 7b: Prewit 3x3 convolution mask horizontal and vertical direction

The Sobel operator uses a variation of the Prewitt operator but with the added emphasis on the centre image pixel.

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Figure 7c: Sobel 3x3 convolution mask horizontal and vertical direction

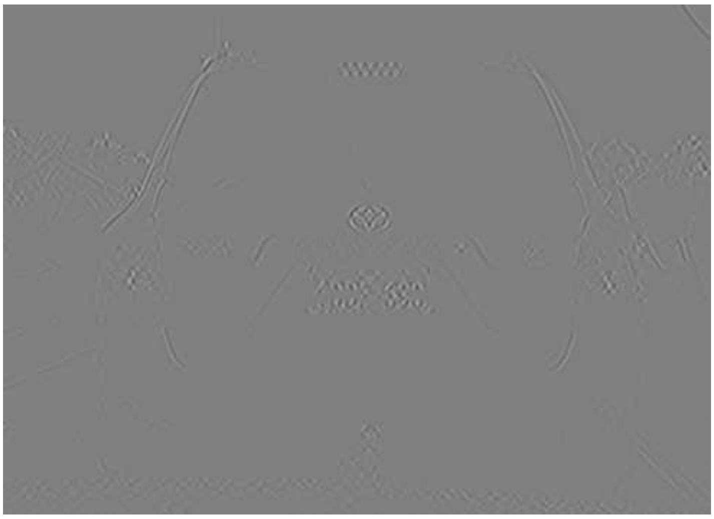


Figure 7d: Sobel edge image using Sobel in the vertical direction

5.6 Canny Algorithm

The Canny algorithm used in this project follows the following procedure:

1) Use a two dimensional Gausissan filter and convolve it with the image to smooth it.
2) From two orthogonal directions differentiate the image.
3) Calculate the gradient direction and amplitude.
4) Perform a non-maximal suppression, which suppresses gradient values that are not local maxima's down to zero.
5) Using thresholding hysteresis from two threshold values to eliminate insignificant edges.

The canny algorithm uses a first derivative Gaussian filter to remove noise in the image, by convolving the filter with the image. The Gaussian filter effectively blurs the image depending on the size of the Gaussian filter, the larger the filter the more the image is blurred which results in smoothing out the small sharp details which allows for the detection of larger prominent edges. While the smaller the filter the less the image is blurred and allows for the detection of smaller fine edges.

$$g(m,n) = G_\sigma(m,n) * f(m,n)$$

Figure 8a: Smoothing the image with a Gaussian filter

Where

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} exp[-\frac{m^2+n^2}{2\sigma^2}]$$

Figure 8b: Gaussian filter

The canny algorithm uses four filters to detect the directions that the edge in the image is pointing at, the four filters detect in the horizontal, vertical and the diagonal directions. Edge detection operators are used to work out the gradients of the image intensity function and the direction of those gradients, these edge detection operators are usually Sobel, Roberts Cross and Prewitt. The resulting angles taken are rounded to one of the four angles that mark the horizontal, vertical and the diagonal angles.

$$M(n,n) = \sqrt{g_m^2(m,n) + g_n^2(m,n)}$$

Figure 8c: Gradient

And

$$\theta(m,n) = tan^{-1}[g_n(m,n)/g_m(m,n)]$$

Figure 8d: Direction of Gradient

Equations are taken from **[27]**

For the final step, the threshold hysteresis is a very important step as it is the only step that helps generate the edges of the image, depending on what threshold values are used, it can make the edge image very detailed or less detailed. The larger threshold value is usually two or three times larger than the smaller threshold value, as an edge point is only considered a valid edge point if the pixel has an edge gradient that is greater than the larger threshold value. That is, we mark it as an edge point when we are absolutely sure it is an edge point. Once a valid edge point has been identified, any pixels that are connected to it which has a gradient value that is greater than the smaller threshold will also be identified as a valid edge point. The resulting image that is produced has nice continuous edges as a result of this edge threshold hysteresis **[24]**.

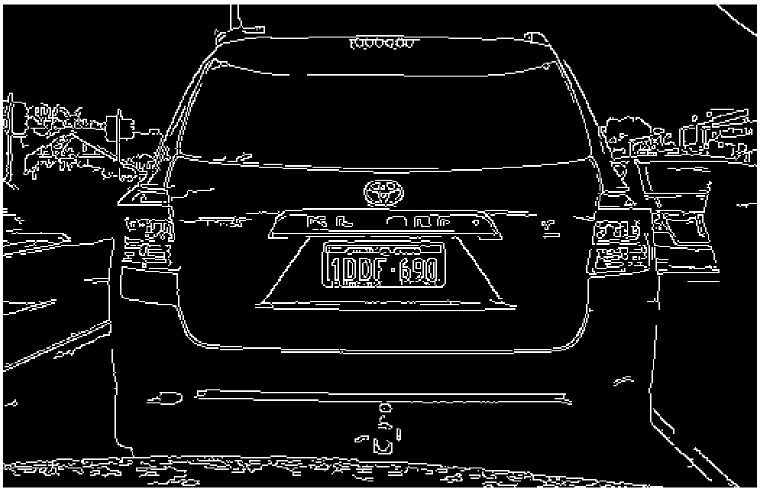


Figure 8e: A Canny edge image

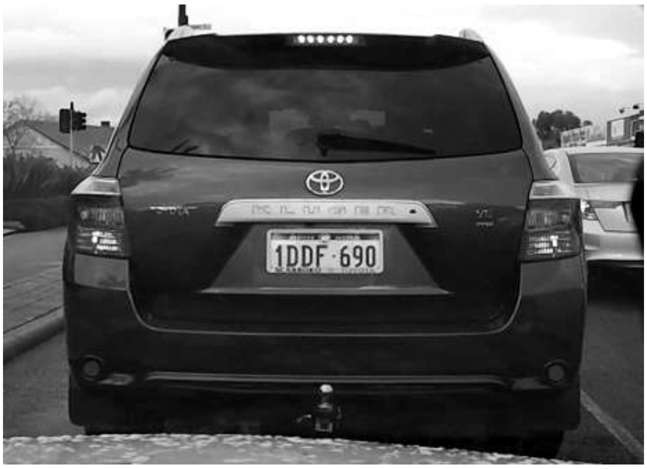


Figure 8f: Original Image

6 The Project

The first few weeks of the project were dedicated to familiarisation with the Eyebot M6 and its libraries. I wrote up a simple interface program that took images from the camera and applied a canny edge detection algorithm to it. The purpose behind this simple program was to give me a feel of what the M6 would be capable of and where potential improvements could be made in image processing were. It was also a good exposure to start programming with the Open CV libraries that specialises in computer vision techniques and algorithms.

6.1 The M6 Interface Program

The purpose of the interface program was to introduce me in how the Eyebot M6 extracts information from the camera images and processes them using Open CV. It also introduced to me the logic loops to have, for the sub menus within the four main buttons on the LCD touch screen. The test program also allowed me to manually send braking and steering commands to the Arduino controller through the serial RS232 port on the M6, which helped a lot in debugging the issues we had for the serial communication between the Arduino and the M6.

The camera sends its output to the FPGA where low level processing is done to it, the image data taken from the FPGA is in an Eyebot Image data structure. To process the image using Open CV functions we must first convert the format of the Eyebot image structure to a format structure that is used by Open CV. Once the image has been used and processed using Open CV functions, to display the results onto the Eyebot LCD screen, it needs to be converted back into the Eyebot Image structure. To do this, there are two functions that I used **[Appendix A]**.

In this program I gave the functionality to change the resolution of the camera output, the option to send out commands via RS232 serial for braking and steering control, the option to display a gray image or an edge image produced by the Open CV function for a canny edge detection algorithm, and the option to freeze the camera feed.

6.2 The PC Test Program

This M6 Test program is the PC version of the vehicle detection program that is to be ported over to the M6 Eyebot controller. It has all the same functionality as the M6 version and is used to compare the performance difference between PC and the M6 embedded system. This test program takes in a recorded video clip and performs stage one of the symmetry based vehicle detection algorithm, and displays the various variables that are used to compare the differences in performance between the two platforms.

I gave the option of choosing which edge detection algorithm to use for the detection, these options are, Canny, Sobel, Laplace, Binary Thresholding algorithms. These are the four edge detection algorithms that Open CV provides which allowed for comparison tests to determine which algorithms would be better suited on the Eyebot M6

6.3 The M6 Detection Program

This program is the ported version of the symmetry based vehicle detection algorithm, which is the main focus of this project. The symmetry algorithm itself is just comparing the symmetry of the image and nothing else, however with this programme I was able to use that information of symmetry and determine if there was an object/vehicle on the road. A detection of an object is only deemed valid if the detection lies within the region of interest in front of the car. Should there be a detected object/vehicle on the road in front of the car, then guestimate the distance of the object in relation to the car via the difference in the number of pixels between the detected point and the base of the image.

The visual display to indicate that an object/vehicle was detected was to draw a box outline indicating the area inside the box contained the object of interest. To draw the detection box around the object/vehicle so that it is of appropriate size, the difference between a chosen constant and the detected point was needed, by using this distance the appropriate size of the box can be determined which follows a linear growth.

The M6 program allows the user to select which forms of assistance they want running in the car, from braking assist that applies the brakes to slow the car down before colliding with the vehicle in front, to steering assist that initiates a steering protocol to avoid a collision with the vehicle in front, or produce audible warning tones to alert the driver about the presence of the vehicle in front. The program uses the distance information, taken from the difference between the chosen constant and the detected point, and determines when it is appropriate to apply the brakes when braking assist is selected, or initiate the steering avoidance protocol when steering assist is selected or produce audible warning tones that the vehicle in front is approaching the car at a close distance.

In Figure 1 below, the location of the detected point is used to determine the size of the detection box for the vehicle, seen in white and the detection point which is also displayed in white on the vehicle itself.

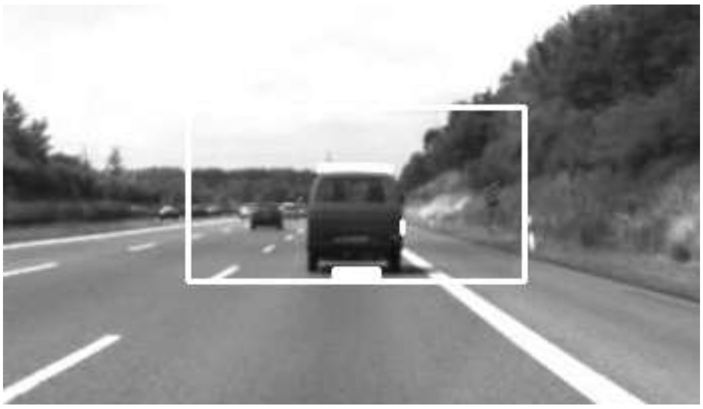


Figure 9a: Detection box

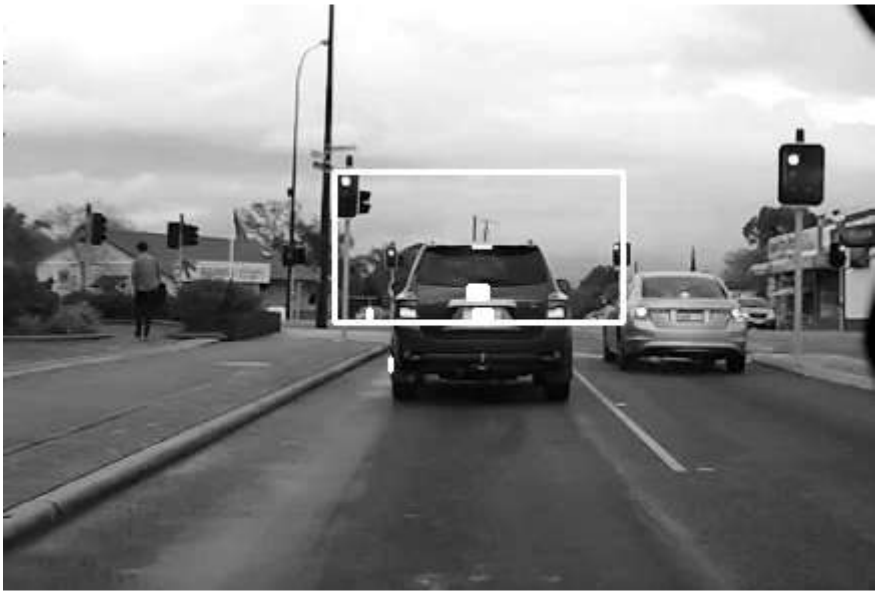


Figure 9b: Detection boxes

In Figure 2 we can see that the detection box did not contain the whole car that it is supposed to detect, this is because the detected point is at a higher place on the car due to the symmetry detected at that row. This is one of the main disadvantages as the symmetry algorithm alone would detect at the wrong place in the region of interest, whereas with stage 2 of the algorithm implemented, the program could check if the detected point is a valid detection by comparing the result with a database.

6.4 Control/Logic Flow

To program the vehicle detection program onto the M6, there needs to be a clear understanding of what is going on in all areas of the project. Below is the general logic behind how the whole system works.

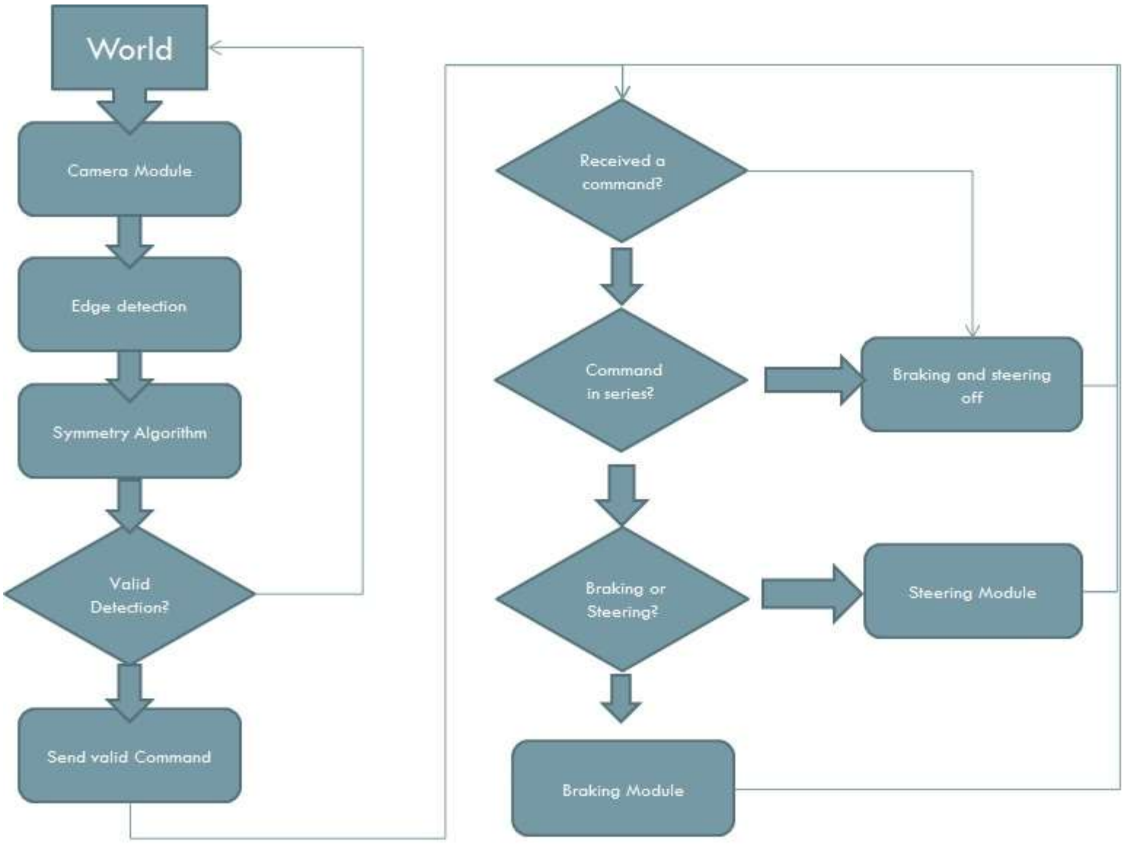


Figure 10: Flow chart of the entire project

The system starts by grabbing image frames from the camera module of the world. The camera module utilizes the FPGA by sending it the raw image and having low level processing done on it to make it usable and displayable onto the Eyebot M6. With the image from the FPGA, the program performs an edge detection algorithm on it giving an image with detailed information about the edge data of the image. From the edge detected image, the symmetry based vehicle detection algorithm then processes from a restricted area called the region of interest where the vehicle will most likely be for a frontal collision. If in the region of interest the symmetry algorithm detects a symmetry set, then it marks that image row as a detected row of interest.

For every detection given it is not efficient nor is it workable to give every detection an output to the Arduino motor controller. As this would result in the motors oscillating between values caused by detections in different places over a short period of time, as well as a large number of detections at the same time in different areas. To lower the large number of detections over several frames so that we can use a sustainable amount of detections that are accurate enough to base the location of the vehicle in the image without picking up random detections in the image as valid detections, it was needed to average out the number of detections in order to smooth out the glitches as the output would oscillate from the large number of detections. I decided to use the median averaging over the mean averaging of the detections, as the mean averaging is prone to be affected by outliers in the detections, whereas the median averaging is not affected by small numbers of outliers. This meant that the accuracy of the detection is established and we reduced the oscillating effect of the output caused by large numbers of detections. The size of the detection box depends on the distance from the bottom of the image; the size of the detection box will be in a linear relation to that distance.

To send the appropriate braking and steering commands and compensate against the curvature of the lens on the camera which distorts the lines on the image making the image lines non-linear, I decided with taking a constant which marks the close proximity of the front of the car in the image and find the difference with the detection point on the image, which I then multiplied by a ratio to compensate as much as I could for the lens distorted image. The result gave a decent usable estimation of the distance to the object/vehicle in front of the car. The commands are send via serial to the Arduino microcontroller where it is checked for validity. It was agreed upon by the mechatronics student, who was working on the motor controllers for the car, and I, that the Arduino is to accept a series of commands before it can proceed to changing the steering and/or brakes as desired. This is to help the system to handle the event that an undetected error in the command was received by the Arduino occurred.

7 Communication with Arduino Platform

Initially it was assumed that the Eyebot M6 would be able to communicate with the Arduino microcontroller via the USB(Universal Serial Bus) slave port on the M6. However upon testing it was found that the USB port was faulty in that the drivers for communicating with it had errors in it, which meant it was not possible to communicate through the USB port on the M6. The only other way to communicate would be through using the RS232(Recommended Standard 232) serial port of the M6 and the TTL(transistor – transistor logic) serial communication port on the Arduino board. To send information from the RS232 port to the TTL serial we would need to use a converter to convert the signals from the RS232 format to the TTL format. To do this I had to create a RS232 to TTL serial converter so that the signals can be converted. To create the converter I used a RS232-to-TTL converter microchip, the MAX232 model **[*See Appendix B*]**, along with a DE9 Male connector to connect the cable and five 1 µF capacitors. All of which were soldered to a prototype PCB board along with a switch that completes the connection to the output of the converter board, this gives me the added functionality to control the command signals going into the Arduino board for the debugging/testing phase.

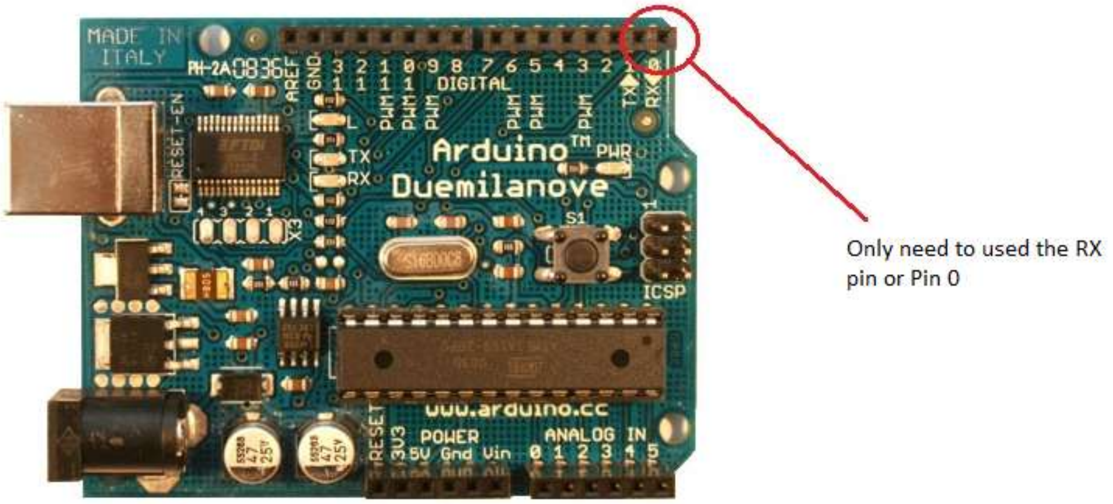


Figure 11: Showing the RX port used

The output of the RS232 – to – TTL converter goes into the RX0 port or pin 0 at the top right of the Arduino board which can be seen in the above figure. Which once processed will send the appropriate PWM signals out from the Digital I/Os to the braking and steering motors **[26]**.

The mechatronics student designing the motor controllers, and I decided that the message commands being sent form the M6 should have a format that is simple to send and that the values sent would have a set range from 0 to 255. We decided that the message format should be of the form:

<message type><message code>

With the message type being restricted to only two types, 'B' for braking and 'S' for steering, and that the message code would be in the range of 0 to 255. Such that to send a brake command would be B127 and a steer command would be S120.

Another area to take into consideration is that inside the vehicle there are bound to be some noise in the cable due to the electronics/radio etc., to solve this we needed a noise resistant way to communicate to the Arduino board, as having noise in the system would possibly mean errors and glitches in the brakes and steering, due to the commands changing values to the motors. To prevent this there were two sets of protocols put into place to reduce the probability of communication error due to noise. To make a channel noise resistant to reduce the effects of noise, there needs to be a noise/error detection algorithm that can detect the errors, which the system can then decide which action to take to handle the error in the data. The system also needs to be able to deal with the off chance that undetected errors could pass through, ie a valid codeword but not the same one sent by the transmitter.

A cyclic redundancy check(CRC) error detecting algorithm was chosen because it is very easy to implement and well suited to both block and continuous data transmissions, and are also good at detecting errors caused by noise **[25]**. For the system to handle the event that undetected errors could pass through to the receiver, it was decided that the Arduino microcontroller would need to receive a constant stream of the same commands before acting upon the commands. This helps in the event that there is an error command in the series of the same commands the error would be ignored by the system. The added bonus as well is that it handles the event that the outputs from the algorithm oscillates between two or three outputs, as stage one of the algorithm tends to be 'jumpy' with its detections. The 'jumpiness' or oscillating outputs of the

Eyebot M6 is a result of when the vehicle or object in front of the car is getting closer and as a result the braking levels are switching from a lower to a higher level. It is at this transition in braking levels that are the cause of the oscillating outputs as the algorithm is picking up more symmetry details on the vehicle or object which are at different points on the image as it gets closer. In the event that there are no commands being passed through or commands are sent but not in series, the Arduino motor controller assumes there are no valid detections, which assumes there is no need for braking, which it then sends the zero braking commands to the brakes **[26]**.

7.1 Braking Control

The braking control logic was designed with consideration that the system would only have one source of input, a single camera that provides vision data about the scene in front of the car. Without a secondary input to extract more data about the area in front of the car it becomes hard to accurately determine the exact location of the object or vehicle in front of the car and the distance between the two. The current set up from the motor controllers give a braking resolution of 256 values to use, which means that there are 256 levels of braking power to select from **[26]**.

It was decided that the best way to determine when the braking protocols should be initiated would be based on where the median value of the detection point is positioned on the pixel rows (vertical direction) of the image. Rather than have a linear braking system in relation to the linear distance between the bottom of the image to the centre of the image, where near the centre of the image is zero braking and near the bottom of the image where the car's front is would be max braking, I decided to simplify it by implementing just four stages of braking, which are 25%, 50%, 75% and 100% full brakes. Due to the lens distortion of the image, due to the curvature of the lens, braking stages 25% and 50% was linear in terms of when the brakes were applied when detection was present, whereas the last two braking stages were closer together to compensate for the lens distortion near the edges around the image. This gave a better response as the car got closer to the vehicle in front, increasing the braking levels appropriately.

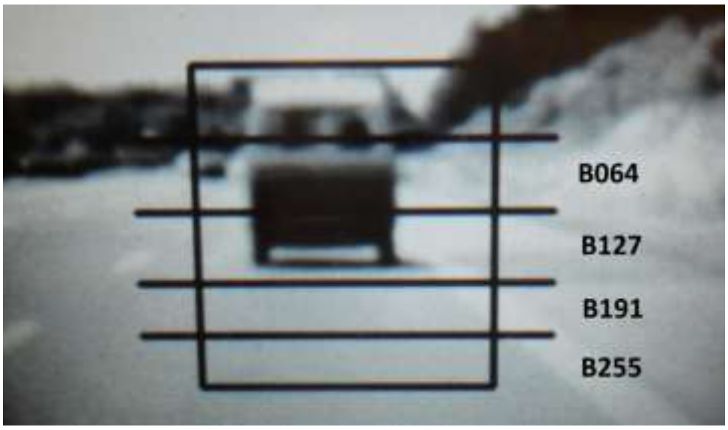


Figure 12: The areas where the braking levels occur

7.2 Steering Control

The steering system was designed to be used to avoid a frontal collision with the object or vehicle in front of the car by applying a set of brakes and steering instructions to steer away from the object or vehicle. In the program I gave the driver the option of choosing which direction to steer to when the algorithm detects a vehicle or object that it is about to collide with, this is to lessen the likelihood of steering onto oncoming traffic and having a head on collision with that. The current set up does not have a smart set up where it is able to decide for itself which direction it should turn to avoid a collision, as this project is supposed to demonstrate that steering with the Eyebot from a single input camera was possible. The current set up of the motor controllers allow for a resolution of 256 values in the steering angles, which is set up so that from the centre there are 128 values to choose from to steer to full lock to the left, and 128 values to choose from to steer to full lock to the right **[26]**.

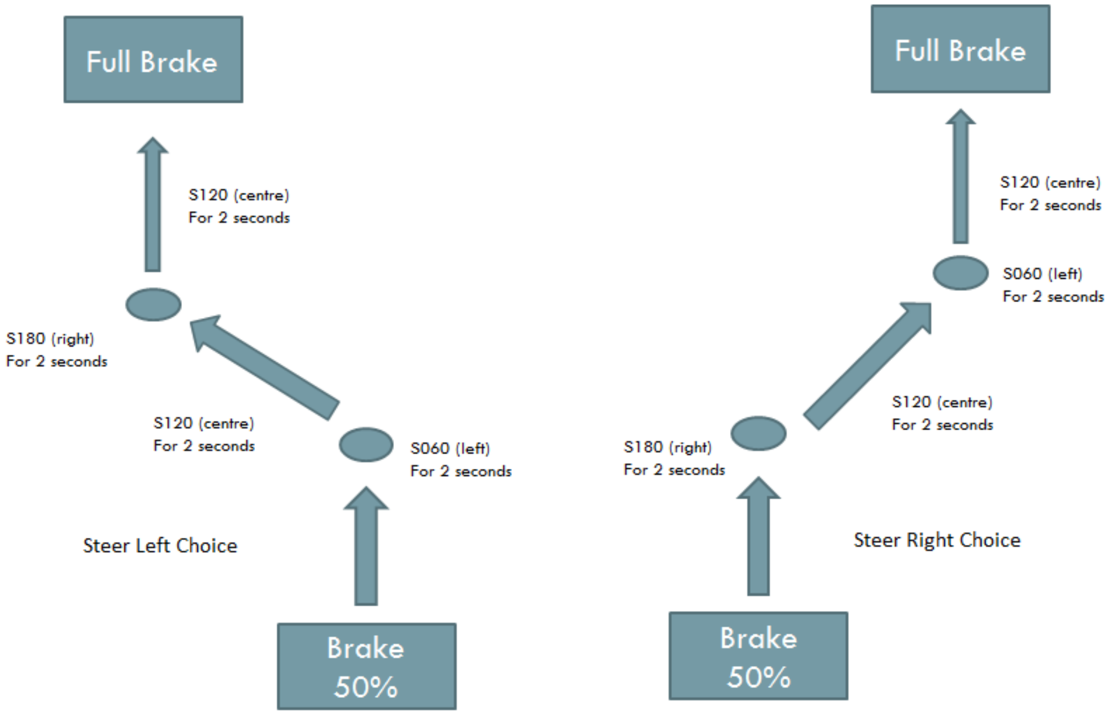


Figure 13: Steering Control flow

In figure 13, it depicts the steering protocol when the program detects a close enough object or vehicle heading towards the car. If the driver selects the option to steer left on the LCD screen the program will turn the car to the left when it detects a vehicle it’s about to collide with following the above control flow.

If the detected vehicle or object lies within the 50% braking range on the image, the program will apply the brakes at 50% and immediately begin to steer left. The program steers the steering wheel at around 60 to 70 degrees to the left and holds for 2 seconds and after the 2 seconds have passed the steering wheel is then turned to its original centre and holds for 2 seconds, this ensures the car is now heading at an angle of around 60 to 70 degrees to the left from its original position. Once it has finished holding the steering wheel at the centre for 2 seconds the program turns the steering wheel at around 60 to 70 degrees to the right and holds for 2 seconds so that the car is changing its direction and after the 2 seconds the program steers the steering back to its centre again and holds for another 2 seconds which the program then applies the full brake to stop the car. This ensures the car is now approximately parallel on the left side, with the vehicle that the car was about to collide with. The same control logic applies to when the driver selects the option of steering to the right for the avoidance selection, but the angles are for the opposite direction.

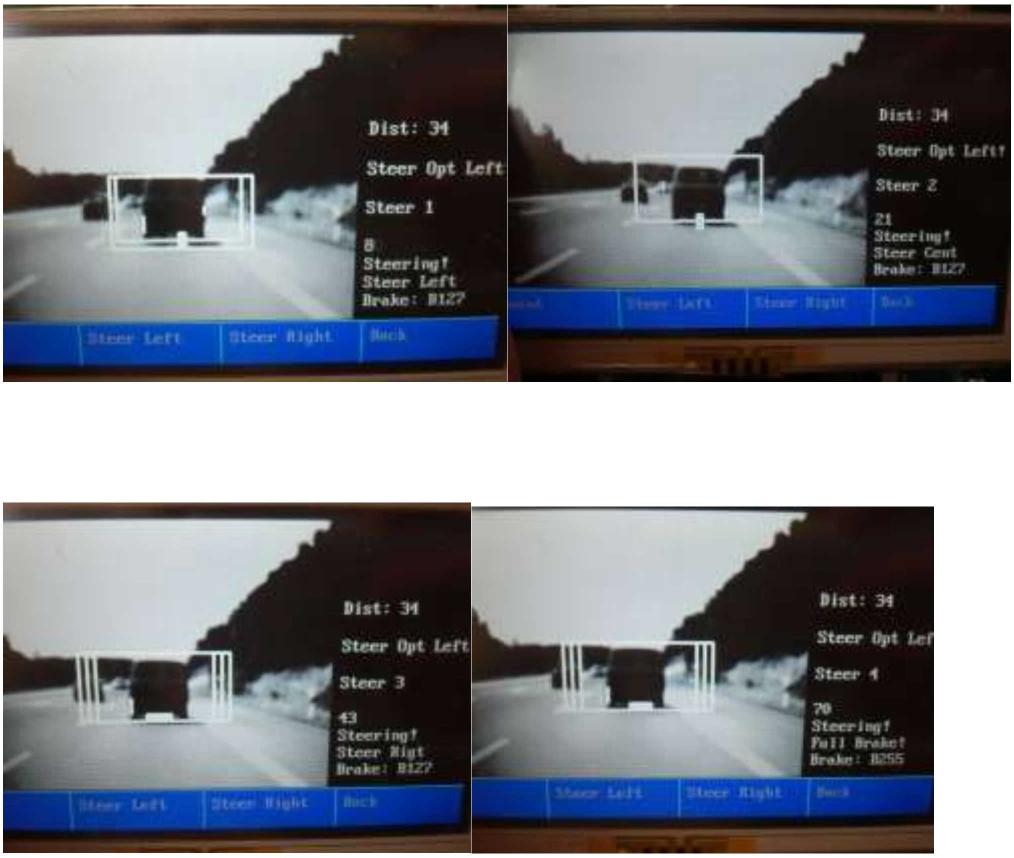


Figure 14: Showing display of steering protocol

8 Driver Assist

The purpose of driver assist technologies is to assist the driver with quick and simple information that are non-intrusive and intuitively easy to depict and understand and clearly visible to the driver. It has been well documented that distracted drivers and inattentive drivers are the main factors in vehicle collisions **[2]**. With this in mind, I designed an audio alert/warning system that alerts the driver when the brakes are being applied, the audio alert increases in tone as the brake level increases. With this audio alert, the driver would be alerted that the brakes are being applied and that the driver should take the necessary steps to avoid an accident, should the driver not intervene the system should initiate the brakes to stop the car and prevent a high speed collision, or steer away if the steering option was selected.

The M6 also provides a visually clear indication that it has detected a vehicle or an object in front of the car, by drawing a distinct white box outline around the location of where that detection is. The size of the white box is related to the distance of the detected point, the further the detected point is to the bottom of the image the smaller the detection box is, while the closer it is to the bottom of the image, ie closer to the car the larger the detection box is. This is to encompass the detected object/vehicle inside the white box so it clearly shows that the vehicle is detected and is directly in front of the car.

The user interface of the program allows the driver the option of selecting to have the steering system active or not. The braking assistance is always active by default and can be turned off if the driver wants. This is to empower the driver in allowing the driver to select the level of assistance that the driver wants, and also allows the driver to take full control of the situation if they want to. There is also a visual status display on the LCD which shows if the brakes and or steering systems are active or not, it will also display the level of braking being applied as well as the steering levels.

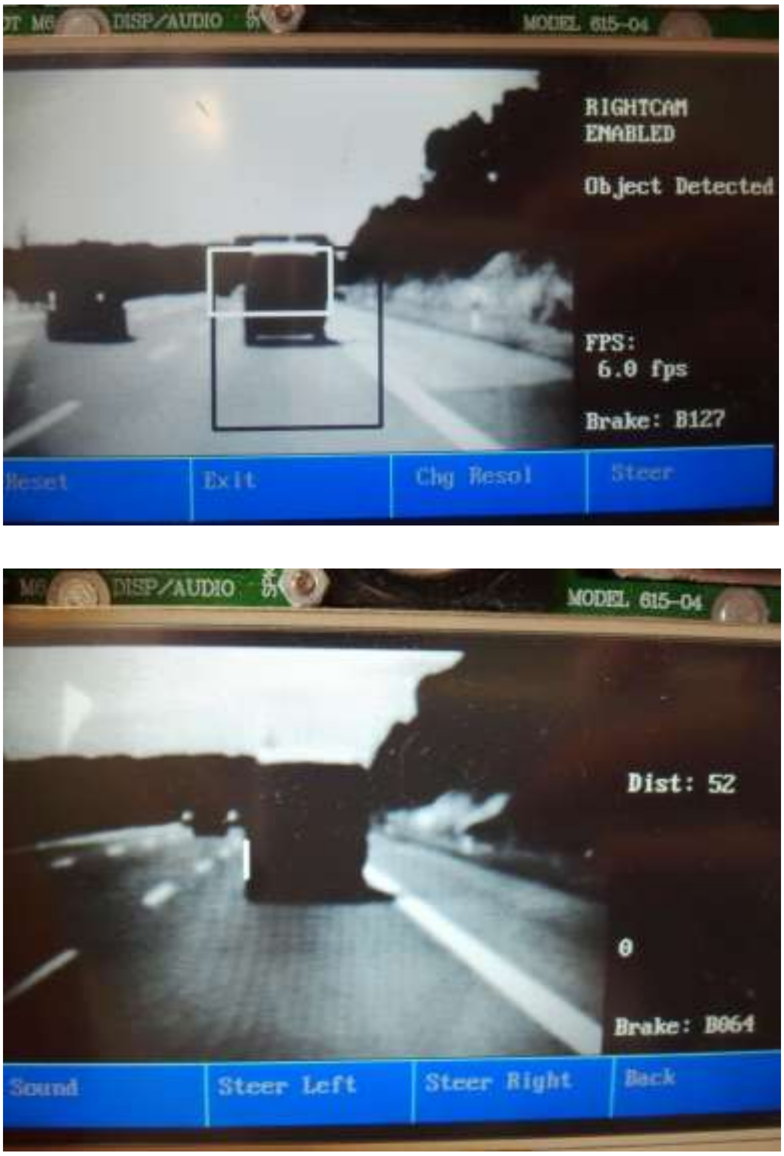


Figure 15: Displaying the different menus

9 Optimization

The Eyebot M6 has very low frame rates when running the image processing via the Gumstix CPU, however if the image processing is done via the on-board FPGA the number of frames able to be processed by the system would be in order of magnitudes better than just via the CPU **[7]**. Currently the FGPA acts as an interface to the camera for the CPU, ie it does low level processing to render the image from the camera, which means that most of the processing of the images are done via the CPU which results in a lower number of frames being processed per second by the system, as the whole process takes an expensive computational time. There is a big performance difference between an optimized process and a non-optimized process, which hence determines the number of frames being able to be processed each second. To measure the benefits of optimization, I decided to measure the differences between full and reduced resolution and their related number of detections, the accuracy of the detections and the frames being processed per second.

To optimize the processing rates of the Eyebot and the PC versions of the same program, a region of interest was introduce to the program to reduce the area of processing from the whole image to a small selected area. The original symmetrical detection algorithm had no region of interest integrated to reduce the area of processing. By reducing the region of interest from the whole image down to the area that is directly in front of the car which is a box area that has 28.5% of the original image width and 37.5% of the original image height, the increase in performance can be seen across all three factors, the number of detections, the accuracy and the frames processed per second. The comparative results can be seen in **Appendix C** where it depicts both PC and Eybot M6 before and after the integration of the region of interest optimization for both full and reduced resolutions.

10 Performance Comparisons

Optimizing by just reducing the region of interest is not enough, as optimizing by selecting which edge detection algorithm would be best suited for the Eyebot M6 without sacrificing the performance speed and accuracy of the system would add a boost to the algorithm's performance. The OpenCV library for computer vision algorithms contains four usable edge detection algorithms, which will be tested and compared with each other to determine which edge detection algorithms produced the best performance on the Eyebot M6 and the PC. To prevent a bias result, there will be two videos to test from, this would reveal which algorithms are the best for each particular video, as both are very different in terms of the environment and lighting. The four edge detection algorithms available to be used by the Open CV library are the Canny, Sobel, Laplace and Thresholding edge detection algorithms.

The results of these comparisons can be seen in **Appendix D**. Since the comparison test involves two videos, video one is depicted with the number 1 next to the names of the algorithms used and the same is done on video two. The parameters for each algorithm used are different from each video as the parameters chosen were deemed to give the best performance result for that particular video. The Eyebot M6 tests were conducted over 10 iterations per video to get a decent average of results for the variables that are being compared.

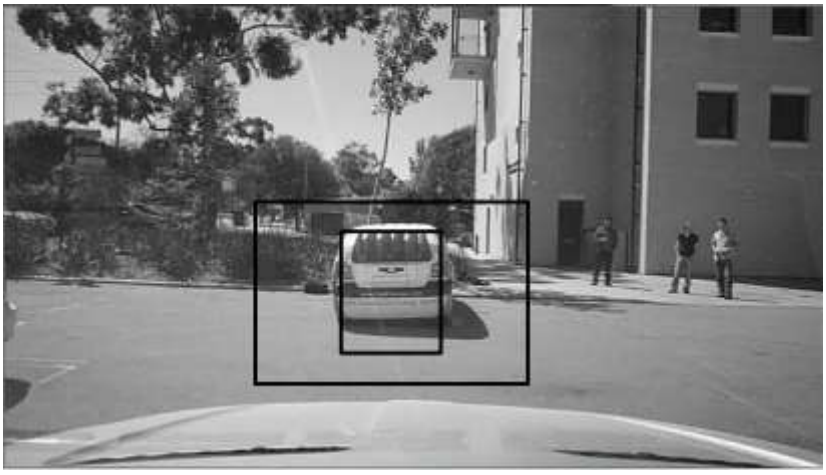


Figure 16a: Video1 with ROI and success area

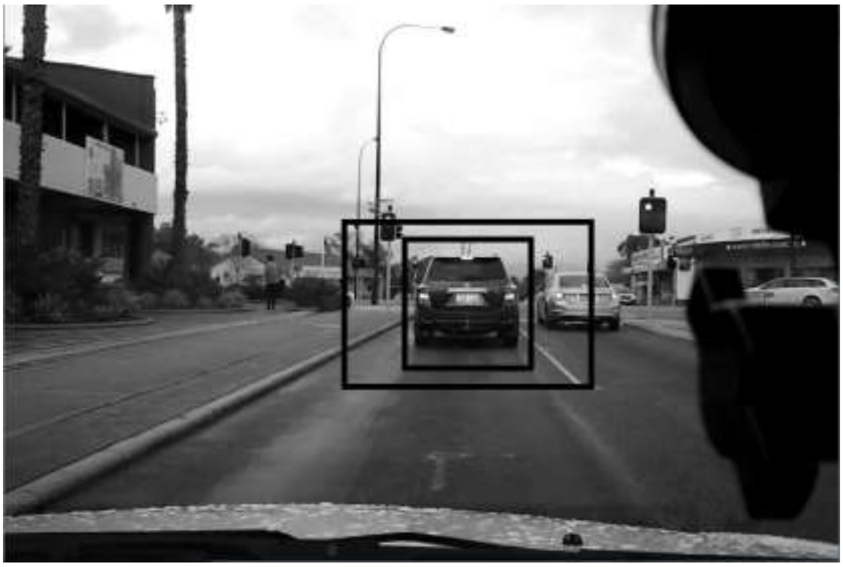


Figure 16b: Video 2 with ROI and success area

10.1 Experiments

From an overview of the results we can come to the conclusion that large numbers of detections does not mean better successful detection rates. For example in the Eyebot M6 case, the full resolution for the Thresholding edge algorithm, the number of detections was evidently large compared to the other algorithms in both videos, with video one reaching 714 detections and video two reaching 1520 while the average detection rates with the other algorithms hovered around the 300 mark. However the percentage of those detections that were successful in detecting the vehicle in the video was in the 30 – 40% range. Compared to Canny which only had a low 65 detections but having 74.44% success in detecting the vehicle in front of it in video one and Laplace with a sizeable 384 detections of which 68.1% were successful in detecting the vehicle in front. Another important fact is that at full resolution the number of detections are a lot higher than the reduced resolution detections, this is due to the fact that at full resolution there are a higher number of pixel rows, which means higher sample intervals in each frame compared with the reduced resolution, which has smaller number of pixels hence lower sample intervals for each frame.

The purpose of this comparison test is to determine which algorithms are most suited to be used on the Eyebot M6. That being said the end results are summarised with stating the top two performing algorithms for each video as follows:

	PC Full Resolution	PC Reduced Resolution	Eyebot Full Resolution	Eyebot Reduced Resolution
Video 1 (1st)	Laplace(3) = 76.2%	Canny = 54.7%	Canny = 74.44%	Canny = 83.67%
Video 1 (2nd)	Canny = 74.2%	Threshold = 51.4%	Laplace(3) = 64.6%	Sobel(0,2,3) = 58.72%
Video 2 (1st)	Canny = 92.44%	Canny = 89.77%	Laplace(3) = 68.1%	Canny = 71.1%
Video 2 (2nd)	Sobel(2,0,5) = 82.5%	Sobel(1,0,5) = 64.81%	Canny = 63.8%	Sobel(0,1,5) = 58.17%

From these summarised results we can see that the Canny edge algorithm is always in the top two performing algorithms. However the downside to such a high accuracy algorithm is that the frame rates per second suffers due to the heavy processing requirements of the canny edge algorithm.

	Eyebot Full Resolution	Eyebot Reduced Resolution
Video 1 (1st)	Sobel(1,0,3) = 7.6 fps	Sobel(1,0,3) = 10.1 fps
Video 1 (2nd)	Sobel(1,0,5) = 7.1 fps	Sobel(0,2,3) = 10.1 fps
Video 1	Canny = 5.9 fps	Canny = 9.1 fps
Video 2 (1st)	Sobel(1,0,3) = 7.6 fps	Sobel(1,0,5) = 10.1 fps
Video 2 (2nd)	Sobel(0,2,3) = 7.3 fps	Sobel(0,2,5) = 10.1 fps
Video 2	Canny = 5.9 fps	Canny = 9.3 fps

11 Advantages + Disadvantages

The major improvements that were accomplished in this project, were made possible by making the Eyebot M6 more selective in its processing. The increase in the frame rates per second was made possible by the introduction of the region of interest concept. By reducing the processing area to a smaller area that is directly in front of the vehicle, the effect is that we have reduced the amount of processing work the CPU has to do and also decrease the number of unimportant detections that would slow down the system and decrease the accuracy of the system. As a result of reducing the processing area of the image, the accuracy of object and vehicle detections that were directly in front of the car improved dramatically which can be seen in detail in ***Appendix D***. The end result of successfully porting over the symmetry based detection algorithm and implementing the system with the steering and braking controllers could be seen in the interview segment on Channel Seven's Today Tonight programme, titled "*Clever Country"*.

As the system has been implemented it is still in its early stages as this is its first implementation into the vehicle system, which comes with a few limitations. One of the major limitations is the current system set up restricts the vehicle's speed to only 40-50km/h, this restriction is so that the speed of the vehicle will allow sufficient data imagery to be processed in the system in real time at a reasonable rate. This limitation is due to the fact that the algorithm is being processed on a slow single core 400 MHz ARM9 controller, which does all the image heavy processing, a faster processor or an optimized programmed FPGA setup would allow for faster processing of the imagery data, which translates into higher vehicle speeds without worrying about the loss of data imagery due to slow processing rates.

Another limitation of the system is that stage one of a three stage vehicle recognition and detection system is not sufficiently accurate at successfully detecting vehicles on the road. This is due to the fact that stage one of the system only does the symmetry processing of the image and stage two compares the data with a database from a machine learning process which passes the results to stage three that tracks the now confirmed vehicle in the image. Without stage two of the detection algorithm, there is no way to confirm the detections detected by stage one are valid vehicle objects, in fact any objects in the region of interest that are detected by stage one are counted as valid objects to avoid colliding with. This is both a good and bad thing, it's a good thing because any objects on the road that is sizeable you would definitely want to avoid colliding or running over it, however without clear confirmation that there is an physical object on the ground in front of the car, symmetrical illusions such as symmetrical shadows on the ground will trigger the detection causing the car to break or steer to avoid that shadow or illusion.

A major limitation of the system that directly affects how effective the entire system can be is the camera system in the Eyebot M6. Depending on how much light there is outdoors, the camera gets easily washed out with sunlight when driving around rendering the image data to be useless. Without any improvements done to it the camera will only be workable when there is a heavy overcast weather, however to solve this problem I added a couple of lens filtering paper to lower the amount of sunlight going through the camera. The resulting image produced by the camera became usable immediately and produced similar results to the indoors tests. However when using the Eyebot M6 during night time driving, the amount of light coming into the camera is insufficient to produce little to no image data, the only solution is to either use an infrared camera or have brighter car lights that flood the area in front of the car with white light, which makes it costly and dangerous for oncoming vehicles.

12 Future Work

As this implementation project has shown, there is still a lot of improvements to be made to the vehicle and embedded system controller before it can be safely utilised by the general public. The user interface on the software needs more refining in terms of visual aesthetics and intuitive displays of information and better user interaction to offer more functionality and give more options to allow the driver access to customize the amount of driving assistance desired. For example, the current system allows the driver the option of selecting the different modes of operations from a basic generic interface layout.

Currently because of the hardware in the Eyebot M6, there are limitations as to how much heavy processing the system can handle before the system becomes slow at processing in real time making it inefficient. There are currently plans to upgrade the Eyebot system, which may include redesigning the entire system layout from scratch. The plans include upgrading the LCD screen, FPGA processor as well as the CPU to a much more current model.

The current camera module has little to no auto contrasting software to compensate in the change in lighting environments. This work needs to be done in order for the system to perform well outdoors with the dynamic changes in sunlight as well as other sources of lighting throughout the day. With the auto contrasting in place there won't be a need for the lens filter to be fixed to the camera lens, which vary in depth accordingly to the amount of daylight in the environment, and should the amount of lighting change, the change of lens filter must be done to get a decent image out of the camera. Another improvement that could be done to the camera, would be the added functionality of auto focusing and auto calibration. Auto focusing would allow the image data coming out of the camera module to be as clear and in focus as much as the camera can allow, this makes processing the data for vehicle detection and recognition much more accurate as the vehicles contours and finer details would be much more easily picked up by the algorithm.

To improve the reliability of the system should the camera system fail, there needs to be other sources of reliable inputs that can compensate and perhaps even give more information to the system when there is an event that disrupts the camera system from giving accurate data about the environment outside the car. For example in the event of a heavy fog on the road, the camera system would fail in giving a fast and accurate response as the camera would not be able to pick up the oncoming vehicle in time for the car to brake sufficiently. A possible solution to this problem would be the addition of an infrared camera mounted to the car, the infrared camera would be able to see pass the visibly thick fog and pick up the heated vehicle in front. The added bonus to adding the infrared camera is its ability to assist the system in detecting vehicles and people during night time where there is a lack of sunlight and insufficient lighting from the car and streetlights. Another good source of input for the visual system would be the addition of laser, sonar or radar systems on the car, though the cost of purchasing such systems would be financially costly, the amount of highly detailed accurate information produced from one these systems, would be more than sufficient to compensate when there is an visual impairing environment event, such as heavy rain, heavy fog/smog and smoke. As these systems are immune to the visual impairing environments, these systems are ideal assisting technologies even though they are more than capable to being the main visual input system.

13 Conclusion

At the end of this project I was able to fully implement stage one of the 3 stage symmetrical based vehicle detection algorithm **[6]**, with a functioning user interface that allows the user to select several options of driver assistance given to the driver when using the Eyebot M6 system. The system has the functionality of giving just an audible warning sound when a vehicle has been detected that is approaching closer to the car. The driver is also given the option of choosing to have auto braking or steering to assist in their driving, both of which at this current time cannot perform sufficiently if the car travels more than 40-50 km/h as it is limited in its processing speed because of hardware limitations. The outputs produced by the system are encoded using CRC error detection protocols to prevent errors appearing at the end of the serial communication cable from happening as the outputs are sent to the Arduino motor controller that controls the braking and steering mechanisms in the vehicle.

By using a camera based visual system, there is the cost benefit of it being cheap and easy to integrate into the system, which makes it an ideal visual based system to choose from for large scale implementation in vehicles. Whereas having a visual system based on more expensive equipment such as lasers and radar would hamper the client with the higher cost for something that could have been implemented with a simpler cheaper camera option.

14 References

[1] *Roy Jurnecka. "Accident Avoidance: Volvo developing new safety systems"*. Internet: http://www.motortrend.com/auto_news/112_news130712_volvo_developing_new_safety_systems/index.html, December 2007 [April 4, 2011].

[2] World Health Organisation 2009, 'Global status report on road safety', Geneva, Switzerland.

[3] Lars Petersson, Nicholas Apostoloff, Alexander Zelinsky, "Driver Assistance: An Integration of Vehicle Monitoring and Control", Proceedings of the 2003 IEEE International Conference on Robotics and Automation , VOL. 2, pg2097, September 2003.

[4] Anup Doshi, Shinko Yuanhsien Cheng, Mohan Manubhai Trivedi, "A Novel Active Heads-Up Display For Driver Assistance", IEEE Transactions on systems, man, and cybernetics –Part B: Cybernetics, VOL. 39, NO. 1,pg 39, February 2009.

[5] Ozgiir Tuncer, Levent Guvenc, Fuat Coskun, Elif Karsligil, "Vision Based Lane Keeping Assistance Control Triggered By A Driver Inattention Monitor", IEEE International Conference on Systems Man and Cybernatics, pg 289, October 2010

[6] Soo Teoh, Thomas Braunl, *"Symmetry – Based Monocular Vehicle Detection System.",* Submitted to Journal of Machine Vision and Applications, 2011.

[7] Bernard Blackham, *"The Development of a Hardware Platform for Real – Time Image Processing"*. Final Year Project Thesis, October 2006. School of Electrical, Electronic and Computer Engineering, The University of Western Australia.

[8] Arduino 2011, Hardware: Arduino UNO. Available from: <http://arduino.cc/en/Main/ArduinoBoardUno> [20 October 2011]

[9] Aye Aye New, Wai Phyo Aung and Yin Mon Myint, "*Software Implementation of Obstacle Detection and Avoidance System for Wheeled Mobile Robot*". World Academy of Science, Engineering and Technology 42 2008.

[10] 3D laser mapping products for environment mapping. Available from: http://www.3dlasermapping.com/ [OCT 2011]

[11]Mobile 3D laser mapping products and services to map streets and render 3D environments for surveying. Available from: http://www.streetmapper.net/brochure/StreetMapper%20-%20General.pdf [OCT 2011]

[12] Sonar mapping product. Available from: http://www.sonartrx.com/blog/ [OCT 2011]

[13] Kane Usher, "*Obstacle avoidance for a non-holonomic vehicle using occupancy grids*" , presented at the 2006 Australasian Conference on Robotics and Automation, Auckland New Zealand, 6 – 8 December 2006.

[14] Alberto Elfes, "*Sonar-Based Real-World Mapping and Navigation*", IEEE Journal of Robotics and Automation, Vol RA-3, NO.3, pg 249-265, JUNE 1987

[15] B.Dai, Y. Fu, T. Wu, "*A Vehicle detection method via symmetry in multi-scale windows*", IEEE Industrial Electronics and Applications, 2007.

[16] T. Xiong, C. Debrunner, "*Stochastic Car Tracking with Line- and Color Based Features*", IEEE transactions on intelligent transportation systems, vol.5, no. 4, December 2004.

[17] G. Y. Song, K. Y. Lee, J. W. Lee, "*Vehicle Detection by Edge-Based Canidate Generation and Appearnace-based Classification"*, IEEE Intelligent Vehicles Symposium Eindhoven. University of Technology Eindhoven, The Netherlands, June 4 – 6, 2008

[18] S. Lefebvre, S. Ambellouis, F. Cabestaing, "*Obstacles detection on a road by dense stereovision with 1D correlation windows and fuzzy filtering"*, Proceedings of the IEEE ITSC 2006, IEEE Intelligent Transportation Systems Conference Toronto, Canada, September 17-20, 2006.

[19] R. Labayrade, D. Aubert, J.P. Tarel, "*Real Time Obstacle Detection in Stereovision on Non Flat Road Geometry Through "V-disparity" Representation*", Proceedings of IEEE Intelligent Vehicle Symposium, June 2002.

[20] K.Huh, J. Park, J, Hwang, D. Hong, "*A stereo vision-based obstacle detection system in vehicles*", Optics and Lasers in Engineering, 168 * 178, 2008.

[21] D. Pfeiffer, U. Franke, "*Efficient Representation of Traffic Scenes by Means of Dynamic Stixels*", IEEE Intelligent Vehicles Symposium University of California, San Diego, CA, USEA, June 21-24, 2010.

[22] Nadra Ben Romdhane, Mohamed Hammami and Hanene Ben-Abdallah, "*A Generic Obstacle Detection Method For Collision Avoidance*", IEEE Intelligent Vehicle Symposium, Baden-Baden, Germany, pg 491-496, June 5-9, 2011

[23] Lei Zhai, Shouping Dong, Honglian Ma, "*Recent Methods and Applications on Image Edge Detection*", In Proceedings of the 2008 international Workshop on Education Technology and Training & 2008 international Workshop on Geoscience and Remote Sensing, pg 332-335, 21 – 22 December 2008

[24] Computer Vision Lecture on Edge Detection, School of Computer Science & Software Engineering, The University of Western Australia.

[25] David Huang, Roberto Togneri "Chanel Codes: Binary Linear and Cyclic Codes". Chapter 2 of Lecture Notes, Digital Communications and Networking. October 2011. School of Electrical, Electronic and Computer Engineering, The University of Western Australia.

[26] Mathew Webster, "*Mechanical Actuation and Low Level Control for a BMW X5 Automatic Safety System*", Final Year Project Thesis, June 2006. School of Mechanical Engineering, University of Western Australia.

[27] *Ruye Wang*, http://fourier.eng.hmc.edu/e161/lectures/canny/node1.html , 20 – September 2004, Computer Image Processing and Analysis (E161) unit. Harvey Mudd College, California. [20 Oct 2011]

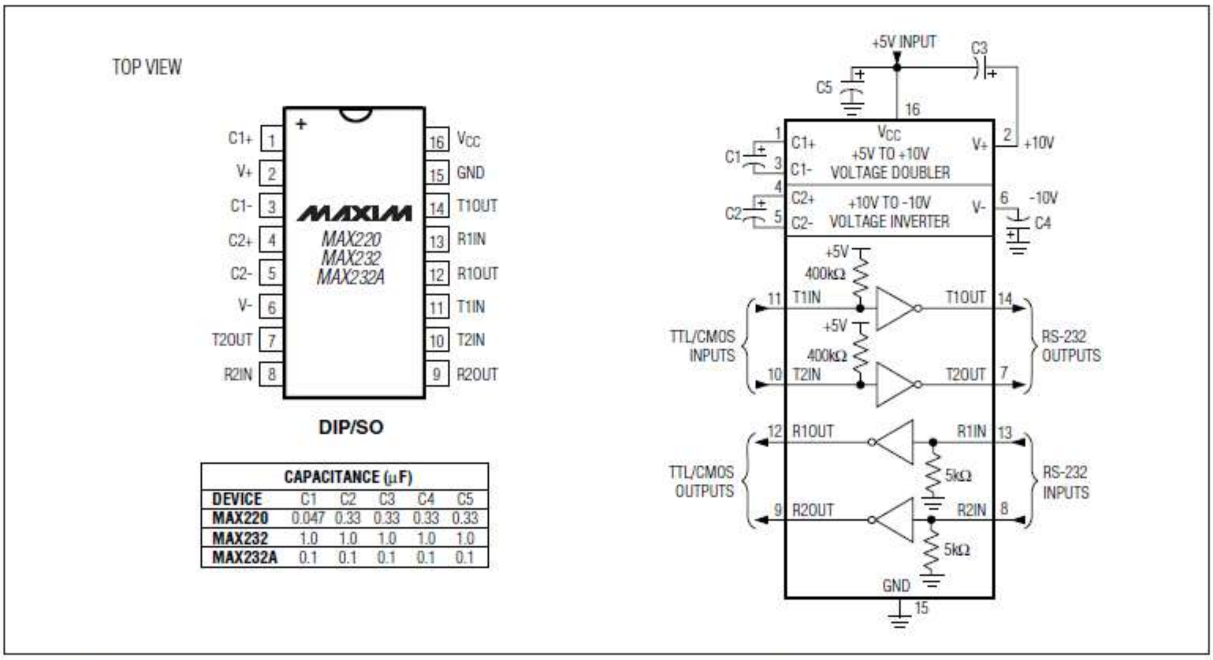
15 Appendix

Appendix A

```
void eyebotImg2IplImg(BYTE *eyeimg, IplImage **iplimg, int QSz)
{
      int height,width,channels;
      uchar *data;

      if (QSz)
            (*iplimg) = cvCreateImage(cvSize(CAM_WIDTH/2,
CAM_HEIGHT/2), IPL_DEPTH_8U, 1);
      else
            (*iplimg) = cvCreateImage(cvSize(CAM_WIDTH, CAM_HEIGHT),
IPL_DEPTH_8U, 1);

      // get the image data
      height    = (*iplimg)->height;
      width     = (*iplimg)->width;
      channels  = (*iplimg)->nChannels;
      data      = (BYTE *)(*iplimg)->imageData;

      memcpy(data, eyeimg, height*width*channels);
}
```

```
void IplImg2eyebotImg(IplImage *iplimg, BYTE *eyeimg, int QSz)
{
      int height,width,channels;
      uchar *data;
      height    = (iplimg)->height;
      width     = (iplimg)->width;
      channels  = (iplimg)->nChannels;
      data      = (uchar *)(iplimg)->imageData;

      memcpy(eyeimg, data, height*width*channels);

}
```

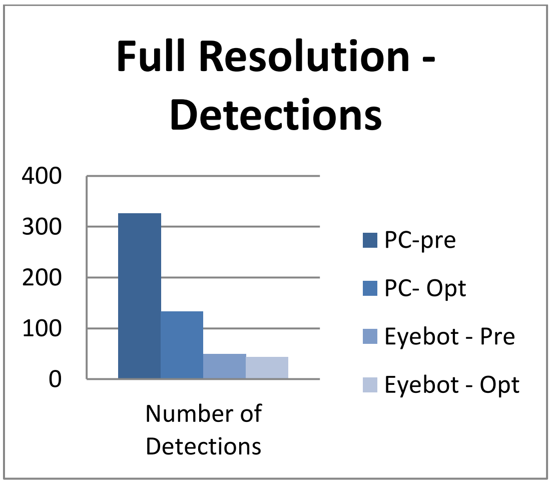
Appendix B

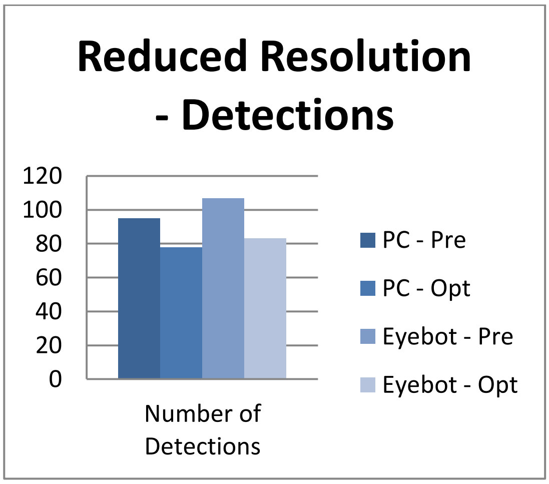


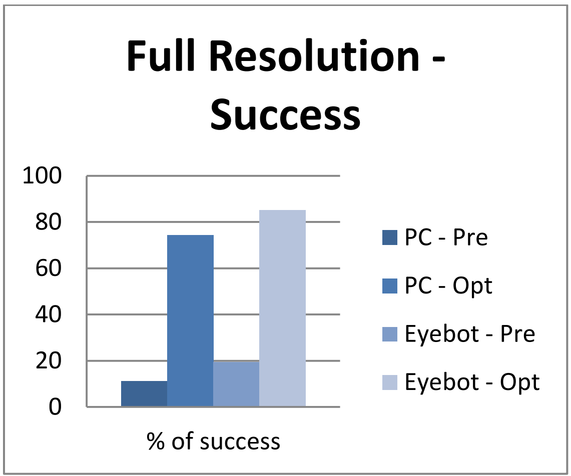
CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

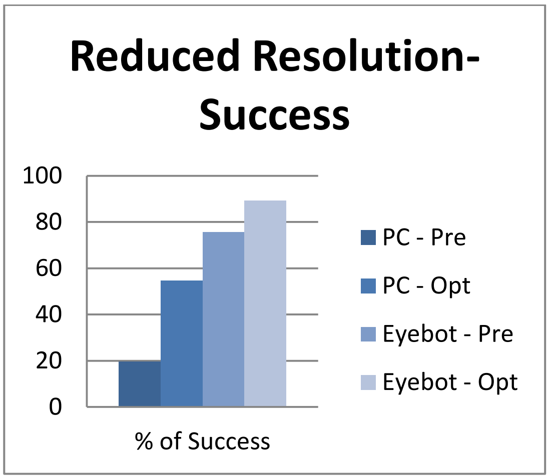
Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

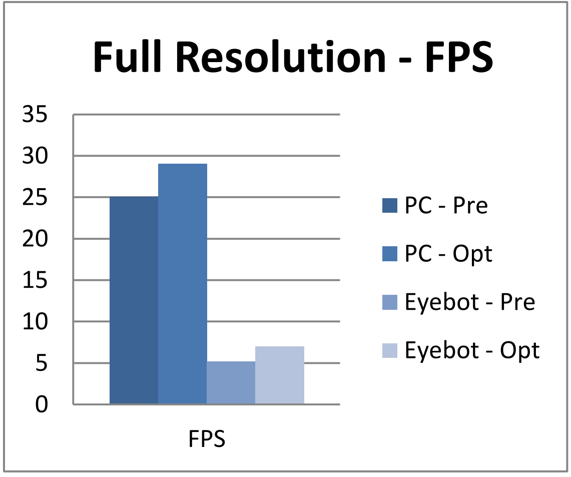
Appendix C

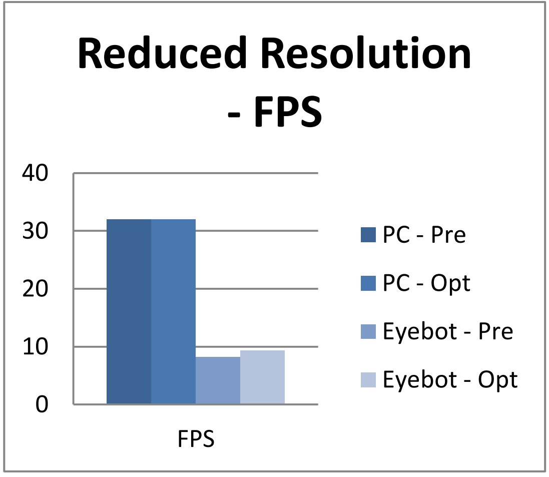




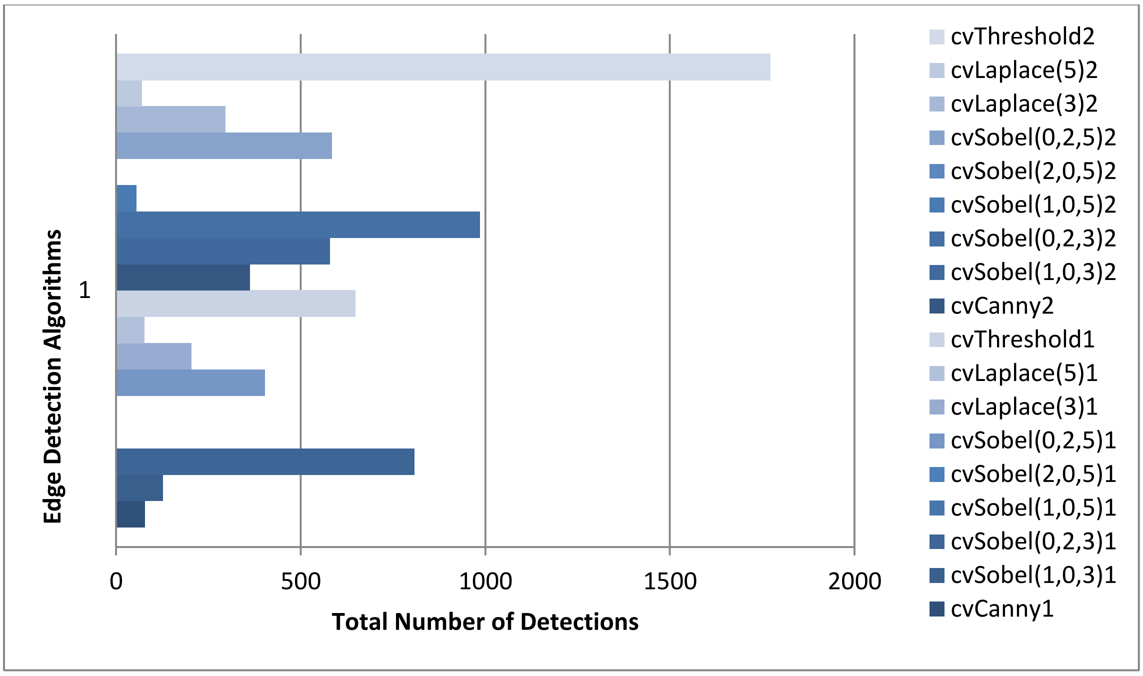

Full Resolution - Success
100
80
60
40
20
0
% of success
PC - Pre
PC - Opt
Eyebot - Pre
Eyebot - Opt


Reduced Resolution- Success
100
80
60
40
20
0
% of Success
PC - Pre
PC - Opt
Eyebot - Pre
Eyebot - Opt

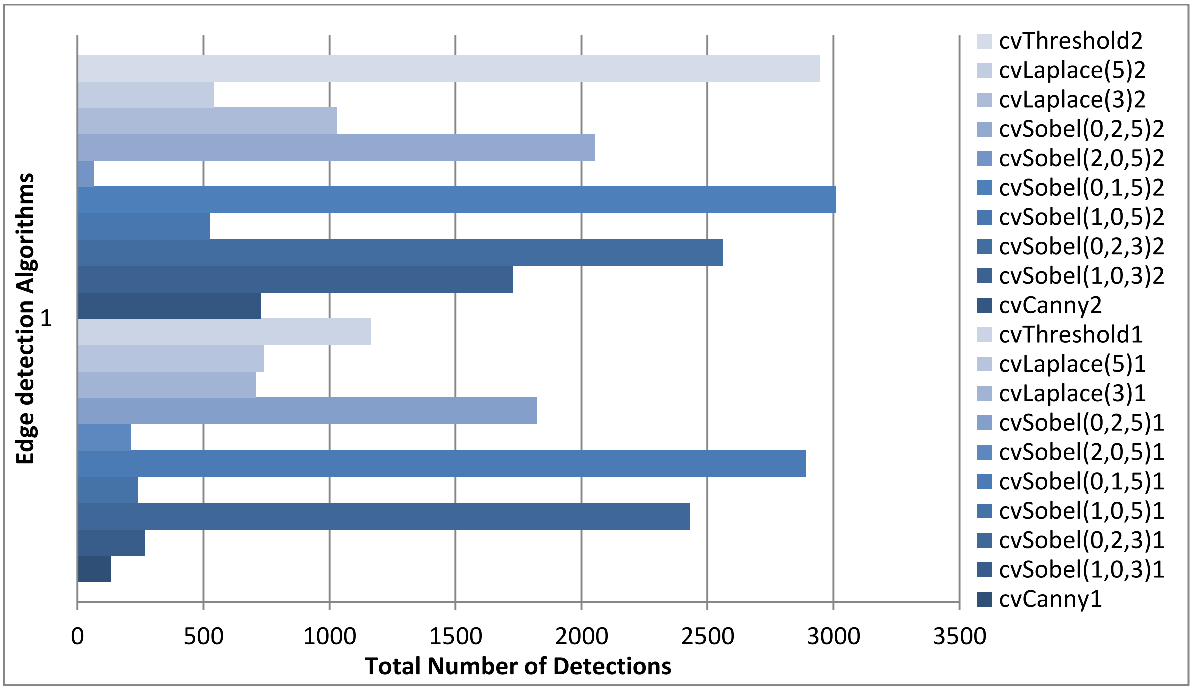

Full Resolution - FPS
35
30
25
20
15
10
5
0
FPS
PC - Pre
PC - Opt
Eyebot - Pre
Eyebot - Opt


Reduced Resolution - FPS
40
30
20
10
0
FPS
PC - Pre
PC - Opt
Eyebot - Pre
Eyebot - Opt

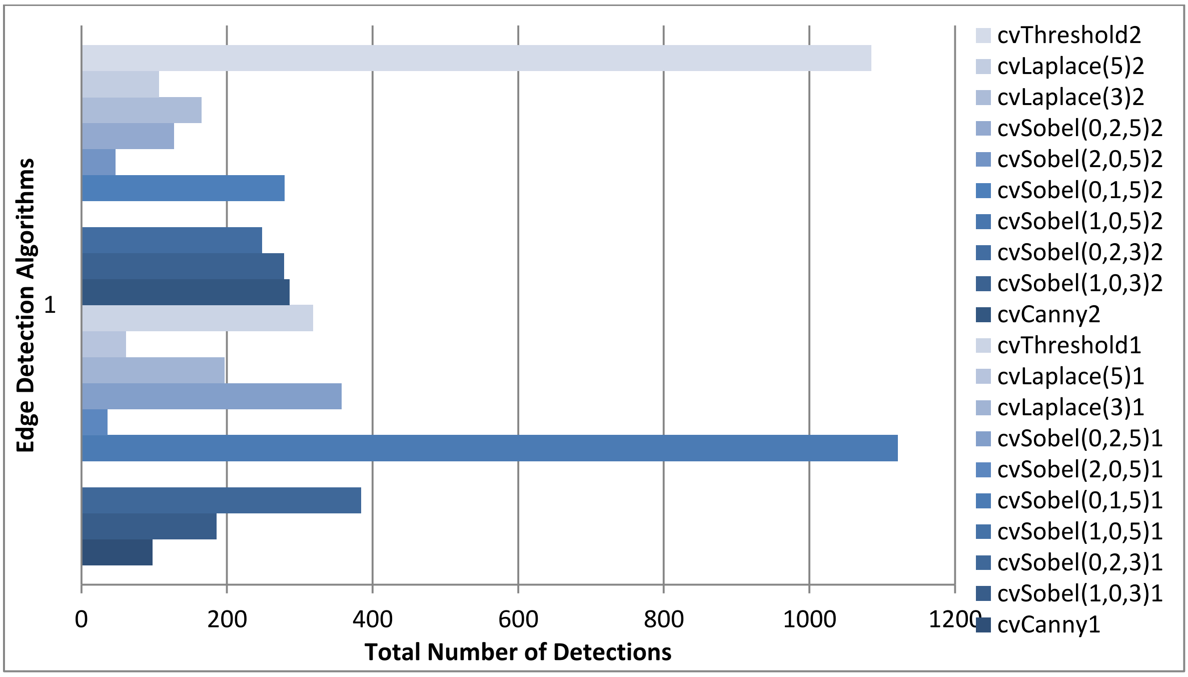
Appendix D



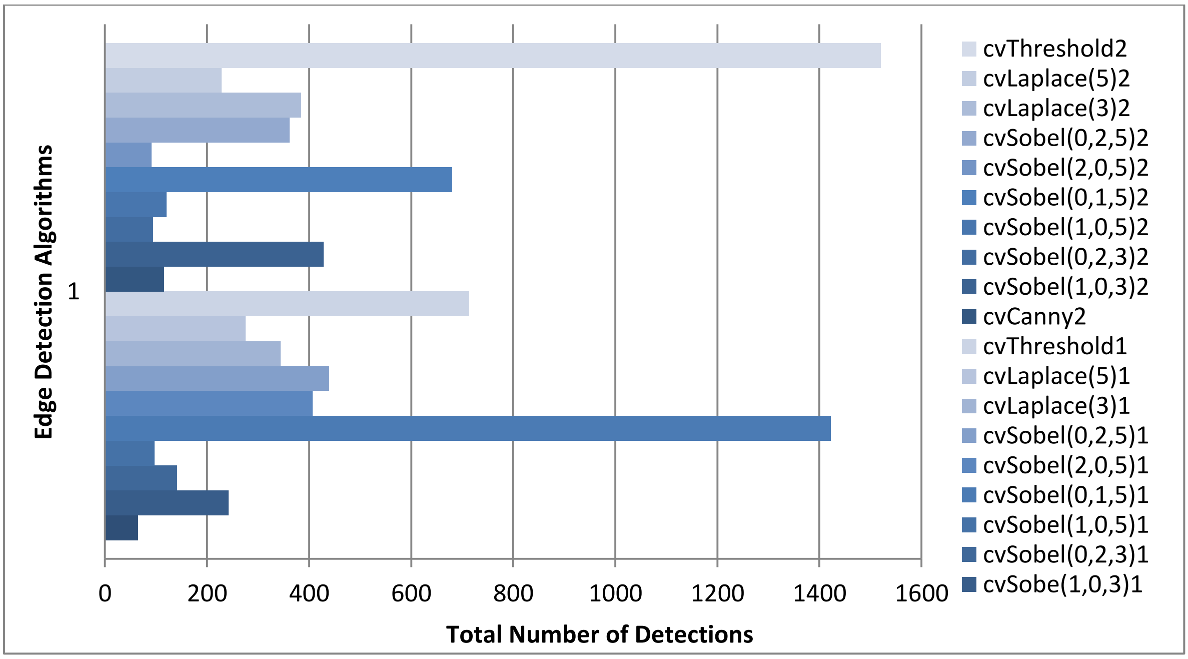
PC – Reduced Resolution



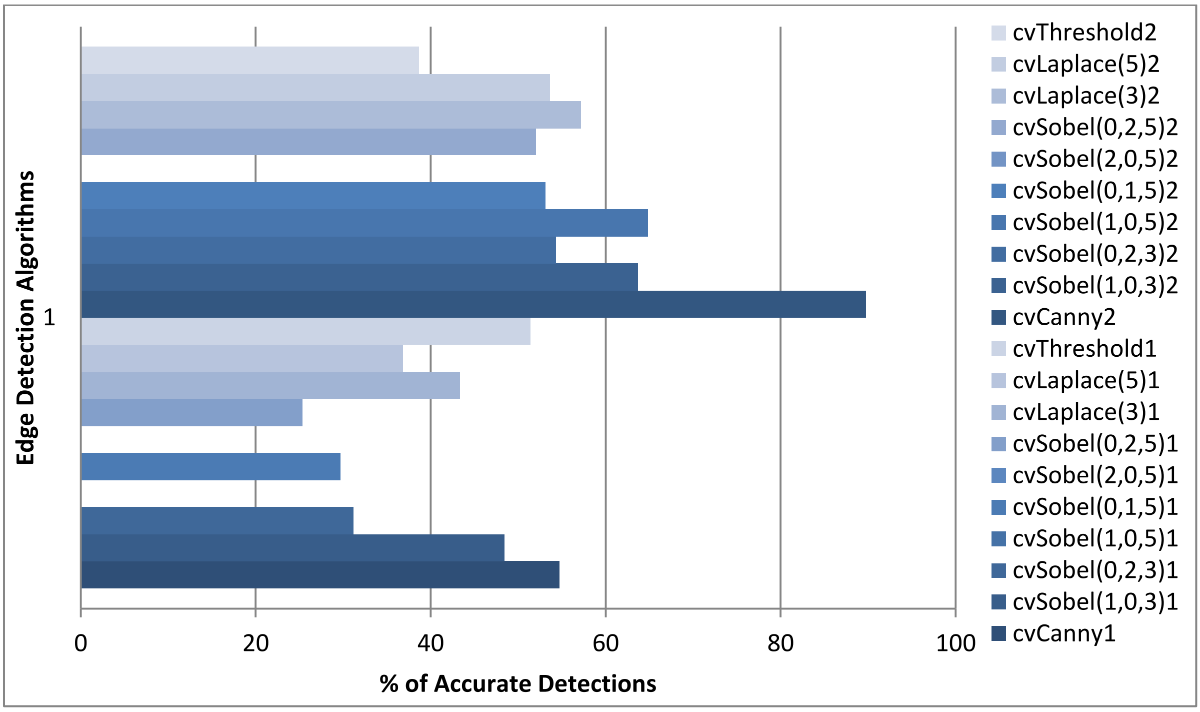
PC – Full Resolution



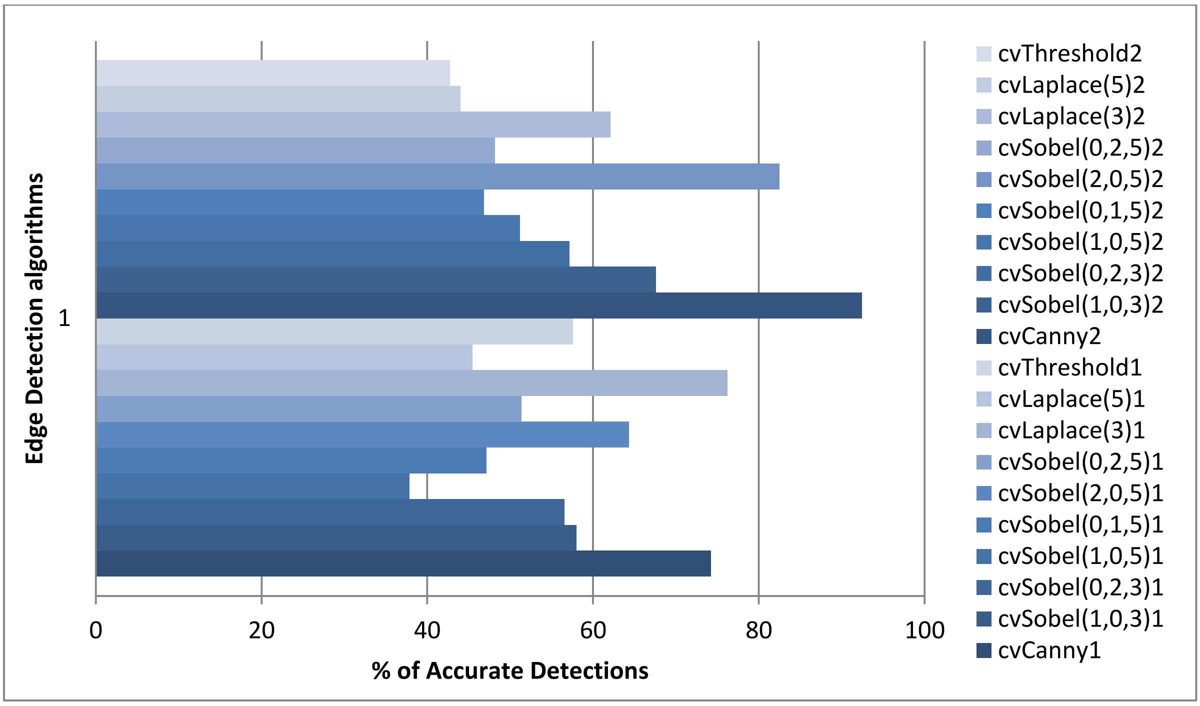
Eyebot M6- Reduced Resolution



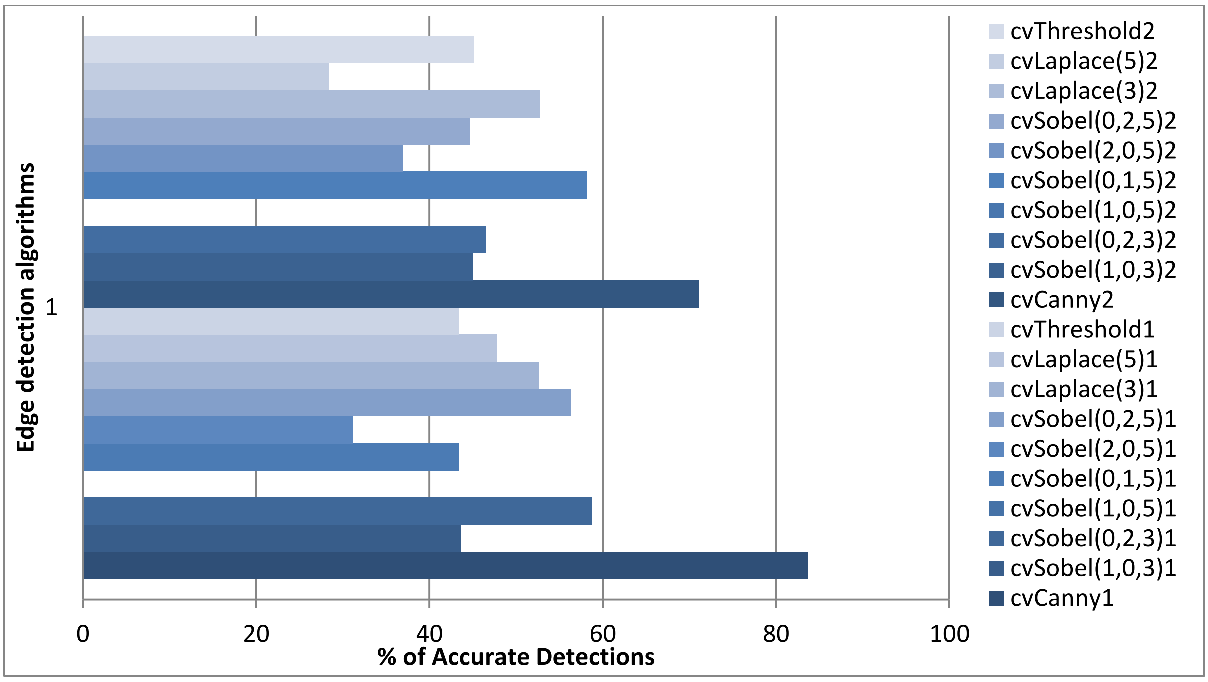
Eyebot M6- Full Resolution



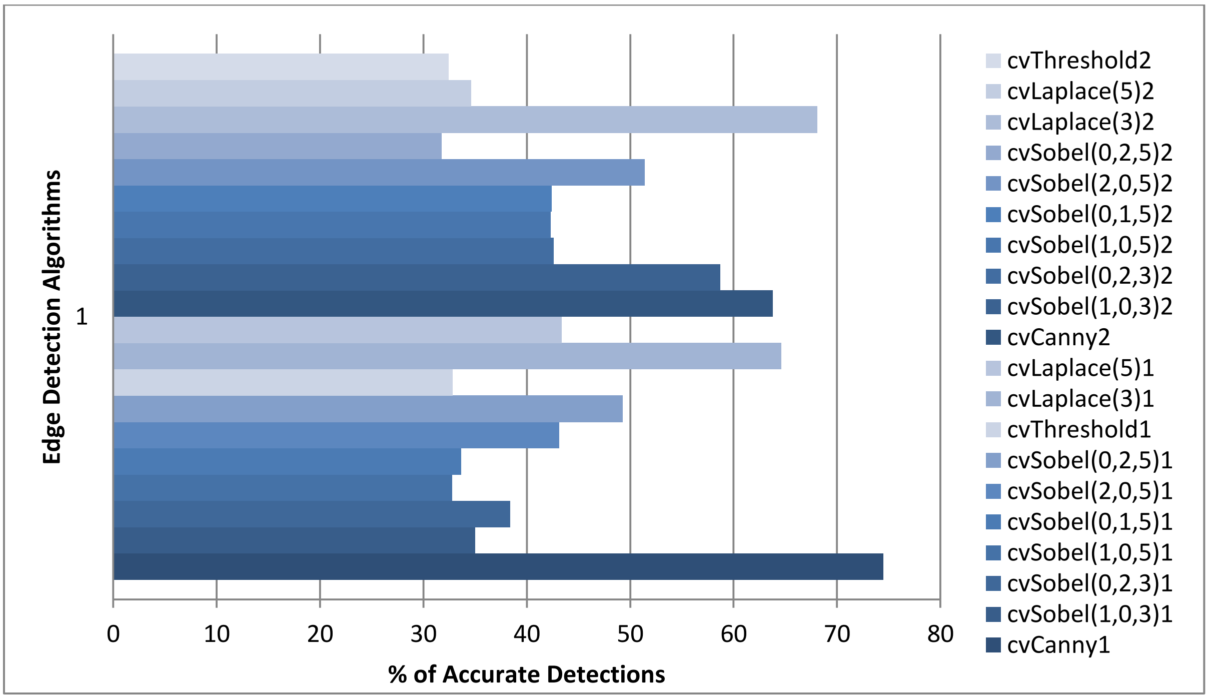
PC – Reduced Resolution



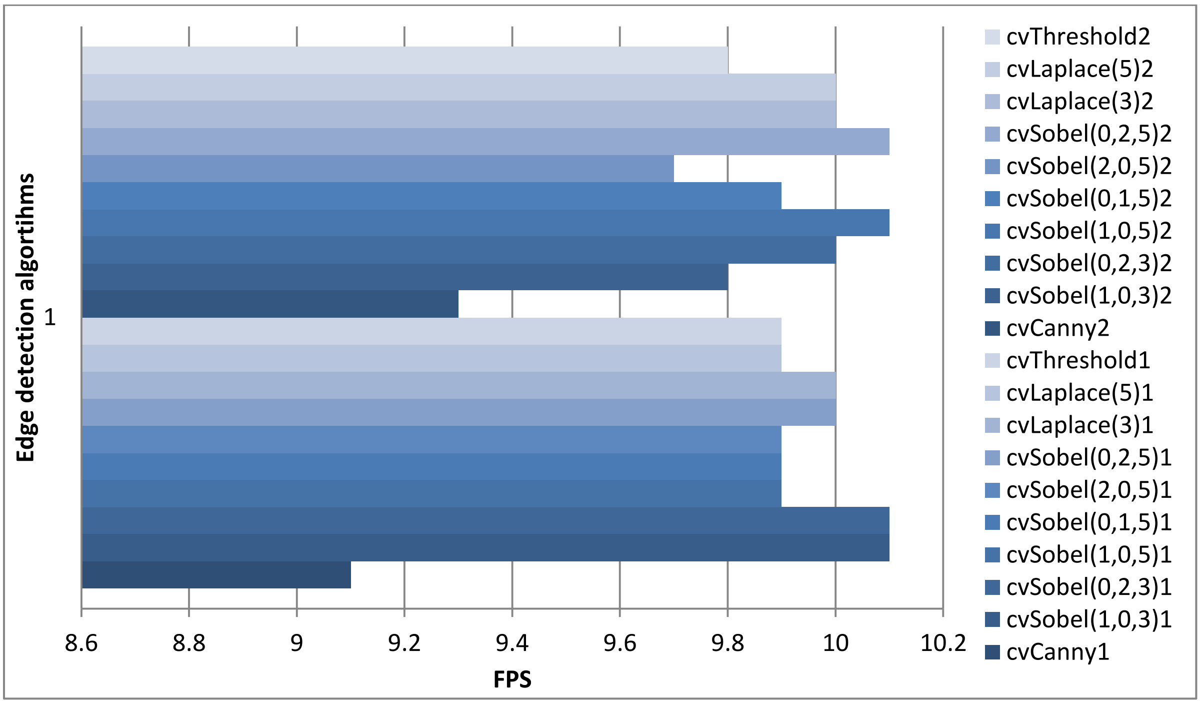
PC – Full Resolution



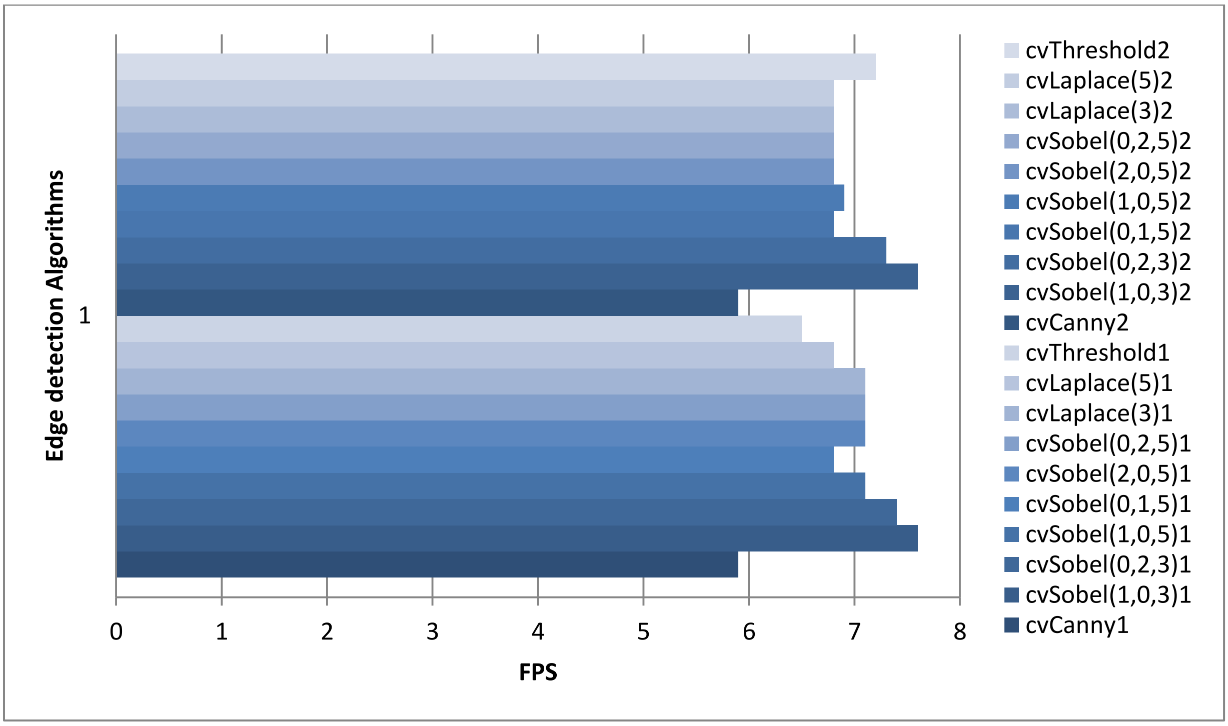
Eyebot M6 – Reduced Resolution



Eyebot M6 – Full Resolution

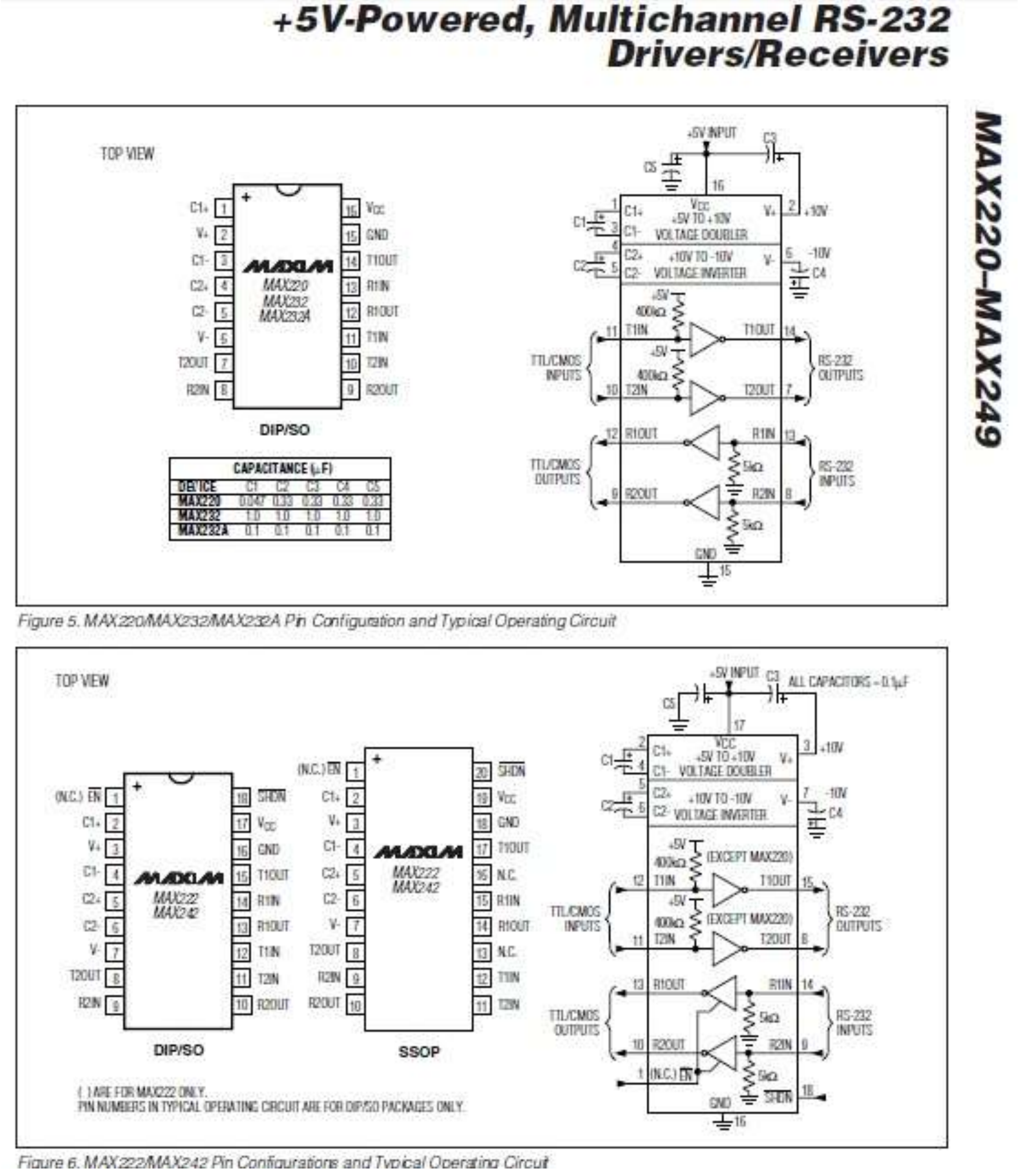


Eyebot M6 – Reduced Resolution



Eyebot M6 – Full Resolution

Appendix E



+5V-Powered, Multichannel RS-232 Drivers/Receivers

MAX220–MAX249

CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

( ) ARE FOR MAX222 ONLY.
PIN NUMBERS IN TYPICAL OPERATING CIRCUIT ARE FOR DIP/SO PACKAGES ONLY.

Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit

Layout of the RS232 - to – TTL serial converter set up